2015

Contrast Pattern Aided Regression and Classification

Vahid Taslimitehrani
Wright State University

Contrast Pattern Aided Regression and Classification

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy

By

VAHID TASLIMITEHRANI
B.S., Azerbaijan University, 2004
M.S., Iran University of Science & Technology, 2007

2016
Wright State University

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

February 19, 2016

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Vahid Taslimitehrani ENTITLED Contrast Pattern Aided Regression and Classification Web BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Guozhu Dong, Ph.D.
Dissertation Director

Michael Raymer, Ph.D.
Director, Computer Science and Engineering Ph.D. Program

Robert E.W. Fyffe, Ph.D.
Vice President for Research and Dean of the Graduate School

Committee on
Final Examination

Guozhu Dong, Ph.D.

Amit Sheth, Ph.D.

Krishnaprasad Thirunarayan, Ph.D.

Keke Chen, Ph.D.

Jyotishman Pathak, Ph.D.

ABSTRACT

Taslimitehrani, Vahid. PhD Department of Computer Science and Engineering, Wright State University, 2016. Contrast Pattern Aided Regression and Classification.

Regression and classification techniques play an essential role in many data mining tasks and have broad applications. However, most of the state-of-the-art regression and classification techniques are often unable to adequately model the interactions among predictor variables in highly heterogeneous datasets. New techniques that can effectively model such complex and heterogeneous structures are needed to significantly improve prediction accuracy.

In this dissertation, we propose a novel type of accurate and interpretable regression and classification models, named as Pattern Aided Regression (PXR) and Pattern Aided Classification (PXC) respectively. Both PXR and PXC rely on identifying regions in the data space where a given baseline model has large modeling errors, characterizing such regions using patterns, and learning specialized models for those regions. Each PXR/PXC model contains several pairs of contrast patterns and local models, where a local classifier is applied only to data instances matching its associated pattern. We also propose a class of classification and regression techniques called Contrast Pattern Aided Regression (CPXR) and Contrast Pattern Aided Classification (CPXC) to build accurate and interpretable PXR and PXC models.

We have conducted a set of comprehensive performance studies to evaluate the performance of CPXR and CPXC. The results show that CPXR and CPXC outperform state-of-the-art regression and classification algorithms, often by significant margins. The results also show that CPXR and CPXC are especially effective for heterogeneous and high dimensional datasets. Besides being new types of modeling, PXR and PXC models can also provide insights into data heterogeneity and diverse predictor-response relationships.

We have also adapted CPXC to handle classifying imbalanced datasets and introduced a new algorithm called Contrast Pattern Aided Classification for Imbalanced Datasets (CPXCim). In CPXCim, we applied a weighting method to boost minority instances as well as a new filtering method to prune patterns with imbalanced matching datasets.

Finally, we applied our techniques on three real applications, two in the healthcare domain and one in the soil mechanic domain. PXR and PXC models are significantly more accurate than other

learning algorithms in those three applications.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

This dissertation becomes a reality with the support and help of many individuals. It would not have been possible without the support of my advisor, Prof. Guozhu Dong. His patience, motivation, and immense knowledge helped me in my Ph.D. study and research. I could not have imagined having a better advisor and mentor for my Ph.D.

I would like to express my sincere gratitude to Prof. Amit P. Sheth, who was more than generous with his expertise and precious time for the past five years. His guidance and support during my Ph.D. encouraged me to work on real-world problems, with the goal of advancing science as well as building my career.

I would like to express my appreciation to my dissertation committee members. I'm so grateful for the productive comments and suggestions of Prof. Krishnaprasad Thirunarayan and Dr. Keke Chen. I will forever be thankful to my former mentor, Prof. Jyotishman Pathak; he has been a tremendous mentor and a great supporter during my internship at Mayo Clinic, which was the most productive internship I've every had.

I also thank my colleagues at Kno.e.sis Center, Explorys, Mayo Clinic and National Library of Medicine: Pramod Anantharam (Kno.e.sis), Lakshika Balasuriya (Kno.e.sis), Olivier Bodenreider (NIH/NLM), Behzad Ghanbarian (University of Texas at Austin), Jason Guilder (Explorys), Qian Han (Kno.e.sis), Ashutosh Jadhav (Kno.e.sis), Pavan Kapanipathi (Kno.e.sis), Zhongliang Li (Kno.e.sis), Maryam Panahiazar (Kno.e.sis, Mayo Clinic), Naveen Pereira (Mayo Clinic), Matthew Pohlman (Explorys), Hemant Purohit (Kno.e.sis), Wenbo Wang (Kno.e.sis), Sanjaya Wijeratne (Kno.e.sis), Tian Xia (Kno.e.sis) and Shaodan Zhai (Kno.e.sis).

Last but not least, special thanks to my wife and best friend, Haleh, without whose love, encouragement and editing assistance, I would not have finished this dissertation. She is the only person who can appreciate my quirkiness and has faith in me even when I do not.

Words cannot express how grateful I am to my mother, father, sister, and brother for all of their supports and prayers.

1

Introduction

Regression and classification are two categories of data mining methods that are being used to extract models to predict future data trends. Such analysis can help us to have a deeper understanding of the dataset in contrast to a simple statistical analysis. Classification models predict categorical class labels while regression models provide continuous valued functions (numerical class label). For example, we can build a classification model that identifies patients at high risk of developing heart failure which may help physicians to make the right decisions to treat patients [Thompson 2015]. Another example can be a regression model that predicts stock prices using the datasets collected from financial markets [Fenghua et al. 2014].

There are two main goals in the design of classification and regression techniques: **Accuracy** and **Interpretability**. Accuracy concerns the ability of a model to make correct predictions while interpretability describes the ability of a model that allows a human being to understand its behavior by inspecting the model's characteristics such as rules and coefficients [Letham et al. 2013]. Interpretability and accuracy are often contradictory issues in the design of classification and regression techniques. The trade-off between accuracy and interpretability of the prediction models is investigated in different domains such as medicine [Johansson et al. 2011]. Accurate prediction models are more complex and opaque while transparent models may perform poorly in terms of correct predictions.

State-of-the-art classification and regression techniques often fail to produce highly accurate and interpretable models, mainly for the following three reasons:

- Datasets have highly complex structures, and there are many multidimensional interactions between predictor variables and the response variable. We call those interactions **Predictor-Response relationships (PR)**. Traditional classification and regression techniques are usually unable to detect and model those interactions. Our proposed methods will rectify this issue by

constructing a small set of pattern and local model pairs which each pair of them is representing a multidimensional predictor response relationship.

- Most of the state-of-the-art classification and regression techniques follow an assumption called "universal model.", i.e., all models shall be applied on all data instances. This has been the case when the classifiers are for use in ensembles. One way to resolve this issue is breaking the barrier of that assumption, and each local model is to be applied only to data instances matching its associated pattern.

- In some datasets, data points are highly **heterogeneous and diverse**. In other words, the data points distribution is a mosaic of multiple multidimensional distinct components. A traditional approach would ignore the multimodal nature of the data and try to model the entire dataset at once. However, building an accurate global model is often a difficult task and the performance is usually poor. Our proposed methodology splits those heterogeneous datasets into a small set of subgroups and specialized highly accurate and interpretable local models represent the behavior of those subgroups.

To give a concrete example, suppose that our goal is to build a binary classifier that identifies patients with the high risk of developing heart failure in one year after the medical exam date. A traditional algorithm such as logistic regression identifies age and blood pressure as the most important factors increasing the risk of heart failure. Consider a relatively elderly female patient (70 years old) with the normal systolic blood pressure (120 mm/Hg) and history of hypercholesterolemia (high cholesterol). Since the patient has the normal blood pressure, logistic regression estimated the low risk of heart failure. However, logistic regression misclassified her risk, and she has been diagnosed with heart failure in 6 months after the exam date; because the importance of other predictor variables such as blood's cholesterol and age are ignored by the effect of blood pressure on the risk. The fact is that, **if** *patient's age* $\geq$ *60* **and** *has the history of high cholesterol*, then blood pressure is not a leading factor anymore. In fact, there is a complex interaction between these predictor variables, and logistic regression is unable to detect the interaction (predictor-response relationship). [Thompson 2015].

To meet the challenges caused by heterogeneity and diverse predictor-response relationships, we need to introduce

- A new type of regression and classification models (a) that can detect and model complex diverse predictor-response relationships, (2) outperforms traditional regression and classification models, and (c) presents the models in an interpretable manner.

- New algorithms for building such regression and classification models.

IF (Age > 60 AND History of Hypercholesterolemia) **THEN** (Follow local model f_p)	
Pattern	Local model

Figure 1.1: Sample of a pattern and local model

In this dissertation, we present a new type of regression and classification as well as new methods for building those new type of models. Our approach deals with heterogeneous and diverse regression and classification problems. Our method uses two key ideas. First, we use the concept of patterns to logically characterize subgroups of data points and then associate patterns to local models as the behavioral characterization of the hidden predictor-response relationships in those subgroups of data points. Second, we incorporate a small set of patterns and local model pairs to form a **Pattern Aided Regression (PXR)** for numerical prediction problems and **Pattern Aided Classification**(PXC) for classification problems. Each pair of pattern and local model presents a specific predictor-response relationship, and a set of pattern and local model pairs reveals a set of diverse predictor-response relationships in a heterogeneous dataset. For example, figure 1.1 is a pattern that logically characterizes a subgroup of patients and a specialized local model f_p associated with the pattern represents the behavioral characterization of the predictor-response relationship in those patients.

PXR/PXC models are also easy to interpret, since they usually use very few patterns and take advantage of interpretable modeling methods such as linear regression and logistic regression to build local models. As we will demonstrate, our experiments show that PXR/PXC models can achieve much more accurate prediction compare to state-of-the-art regression and classification models.

We also propose two algorithms called **Contrast Pattern Aided Regression (CPXR)** and **Contrast Pattern Aided Classification (CPXC)** to build PXR and PXC models, respectively. Rather than trying to find a prediction model that simply minimizes the training errors, we attempt to identify "large error" instances whose a traditional approach makes large prediction error. Next, we use contrast pattern mining to characterize regions of the data space that mostly contains large error instances. Finally, we tailor dedicated solutions for those specified regions. We refer to these solutions as local models. This dissertation also introduces several quality measures and techniques to improve computational efficiency.

We designed extensive and systematic sets of experiments that demonstrate CPXR and CPXC consistently outperform state-of-the-art traditional regression and classification techniques. We used around 70 benchmark datasets and more than 20 synthetic datasets in our experiments. We investi-

gated the impact of CPXR and CPXC's parameters on the performance. Overfitting is also examined and compared with other competing methods.

Unlike traditional classification and regression techniques, CPXR and CPXC let the data determine which parts of the data space require a specialized local model, and avoid a needlessly elaborate partitioning when one is not needed. The idea of characterizing large error instances using a set of contrast patterns has many advantages, in addition to improving the accuracy of the final model. In particular, CPXR and CPXC make model's errors more interpretable. This is because misclassified data points are more likely to have some common properties, and contrast patterns characterize those properties in an interpretable manner. Observing patterns make easier for a human to figure out what went wrong.

Although a range of classification algorithms, such as decision tree, random forest, and support vector machine, have been developed and applied to many applications, imbalanced data classification has encountered a serious difficulty to most classification algorithms. In our research, we adopted CPXC to handle imbalanced dataset classification problems and introduced a new algorithm called **Contrast Pattern Aided Classification on Imbalanced Dataset (CPXCim)**.

So far, we applied CPXR and CPXC methodologies on three real applications: 1) outcome prediction of Traumatic Brain Injury (TBI) patients using CPXC [Taslimitehrani and Dong 2014]. 2) risk prediction of heart failure patients in 1, 2 and 5 years after the heart failure using CPXC[Taslimitehrani et al.] and, 3) prediction of Soil Water Retention Curve (SWRC) and Saturated Hydraulic Conductivity (SHC) using CPXR [Ghanbarian et al. 2015].

The major contributions of this dissertation are as follows:

1. It articulates the diverse predictor-response relationship phenomenon.
2. It introduces a novel type of regression and classification models (PXR/PXC).
3. It introduces a novel regression and classification method (CPXR/CPXC) to build accurate and interpretable PXR/PXC models.
4. It presents the adoption of CPXC to handle imbalanced dataset classification (CPXCim).
5. It introduces a novel perspective on using patterns to assist classification learning.
6. CPXR and CPXC can be used in analyzing prediction errors or misclassification of traditional approaches.
7. CPXR and CPXC can also be used as a novel methods to analyze heterogeneous datasets.

In the rest of this dissertation, chapter 2 discusses related works. Chapter 3 presents all required preliminaries. In chapter 4, the PXR concept is defined, and then CPXR algorithm is discussed. Chapter 5 is dedicated to PXC concept and CPXC and CPXCim algorithms. Chapter 6 presents three different applications of CPXR and CPXC methodologies: two applications in healthcare and one in soil mechanics. Finally, we conclude with a summary of this dissertation.

2

Related Work

The focus of our research is in the broad area of regression and classification methods and it is not realistic to discuss all related studies here. Therefore, in this section, we will take a deeper look at the ones that are closer to our research. The related works belong to four main groups: numerical prediction methods, classification methods, classification methods on imbalanced datasets and contrast patterns.

2.1 Numerical Prediction Methods

By far, the most popular approach for numerical prediction problems is regression [Galton 1886]. Regression is a modeling technique to find the relationship between one or more predictor variables and a numerical response variable. There are several types of regression including linear, non-linear, piecewise, and generalized linear regression. Linear regression is the simplest type of regression. [McCullagh 1984]. We cannot give a fully detailed treatment of numerical prediction techniques. Instead, this section provides an introduction to the most related techniques.

2.1.1 Piecewise Regression

A regression method that we would like to discuss is piecewise or segmented linear regression (PLR) methods. In the simplest case (one predictor variable), the predictor variable is partitioned into intervals, a separate line fits to each interval and line segments are joint to each other at points called breakpoints [Toms and Lesperance 2003]. It is also possible to fit non-linear models (smooth transition) for each segmentation [Seber and Wild 2003]. Hudson in [Hudson 1996] presented a method to find the overall least squares solution when a completely fitted curve consists of two or more sub-models. However, an important limitation in Hudson method is that the overall model is continuous at each breakpoint. McZGee et all in [McZgee and Carleton 1970] outlined a method that

uses hierarchical clustering to cluster the data instances into partitions that represent the individual regimes of the piecewise regression and apply linear regression to each of them.

PXR models are more general than PLR models. In the literature PLR models typically involve intervals on just one predictor variable. While several papers in the literature mentioned the possibility and desirability of PLR models using intervals on multiple variables, we were not able to find any study that presents an effective algorithm for building such PLR models, perhaps due to the complexity of the problem. Another critical difference is that the PXR models also allow instances to satisfy multiple patterns, and they use weights to combine predictions by local regression models. From a computing perspective, the CPXR algorithm provides a systematic and effective method to search for a desirable pattern set to represent high-quality PXR models. Experiments show that PXR models are more accurate than PLR models.

2.1.2 Multiple Adaptive Regression Splines

Multiple Adaptive Regression Splines (MARS) is a nonparametric regression method without any assumption about the functional relationship between predictor and response variables [Friedman 1991]. MARS is suitable to apply to prediction problems with the numerical response variable. The basic idea of MARS is a divide-and-conquer strategy that splits the data space into regions, and each region has its own regression models. In another word, MARS is similar to the piecewise regression that defines a set of breakpoints and then learns a separate simple regression model for each region. MARS model is in the form of $y = \beta_0 + \sum_{i=1}^{n} \beta_i H_i$, where β_0 is the intercept and $\beta_1, ..., \beta_n$ are regression coefficients. H_i (spline basis functions) can have two forms: 1) a hinge function in the form of $max(0, x - c)$ or $max(0, c - x)$ where c is a constant number. 2) production of two or more hinge functions that generate a rectangular partition in data space and fit a regression model in each partition. MARS is known to handle high dimensional numerical prediction problem [Hastie et al. 2001].

2.1.3 Gradient Boosting Method

Gradient boosting is the regression version of boosting technique [Friedman 2002]. The main idea behind the gradient boosting method (GBM) is generating fitted models iteratively, to maximizing the correlation between the fitting models and the negative gradient of the loss function. The user can choose the loss function, but the squared residual is one of the common loss functions used in different applications [Natekin and Knoll 2013]. The flexibility of GBM in customizing loss function introduces a lot of freedom into the model design.

A key difference between GBM and our method is that GBM treats all incorrectly predicted

data instances in a uniform manner while CPXR focuses on pattern-defined data groups that have accurate prediction models correcting errors of a given baseline model.

2.1.4 Bayesian Additive Regression Trees

Bayesian Additive Regression Trees (BART) is a Bayesian approach for numerically-valued function estimation using regression trees. Regression trees are based on the recursive partitioning of the predictor space into a set of hyper-rectangles in order to fit an unknown function. BART can be considered a sum-of-trees ensemble, with a new estimation approach using a Bayesian probability model [Chipman et al. 2012]. The main idea of BART is enriching Bayesian model averaging by overweighting a prior that regularize the fit. Experimental results show BART is outperforming neural networks [Specht 1991], random forest [Breiman 2001] and boosting [Freund and Schapire 1997]. Often using hundreds of decision trees, BART models are hard to interpret.

2.2 Classification

Classification is a task of predicting a categorical label such as “yes” or “no” for the marketing data, “high risk” or “low risk” for the medical data, and “safe” or “risky” for the loan application data. There are many popular classification techniques such as decision tree [Quinlan 1993], naive Bayes classifier [Han et al. 2011], support vector machine [Vapnik 2013] and neural network [Specht 1991]. In this section, we provide more details regarding the most related classification techniques.

2.2.1 Recursive Partitioning

Recursive partitioning creates a tree to classify instances in the training dataset by splitting the dataset into subsets based on several discrete response variables. The hierarchical tree is a common way to represent the recursive partitions. Each of the leaves represents a final partition of the dataset. Recursive partitioning methods have been studied since the 1980s. The most known methods of recursive partitioning are C4.5 and Classification and Regression Trees (CART). Overfitting is the common criticism of these methods. To overcome this problem, ensemble learning methods such as Random Forests has been proposed.

2.2.1.1 Decision Trees

ID3 [Quinlan 1986], C4.5 [Quinlan 1993], and CART [Breiman et al. 1984] are three of the oldest decision tree algorithms. The decision tree algorithms start with the entire training dataset in the root and on each step iterate through every predictor variable and calculate the entropy of that

variable. In the next step, the algorithms select the variable with the smallest entropy value. The dataset is then partitioned by the selected variable to generate subsets of the training data. The algorithms continue to iterate on each subset until some stopping criterion is met. Some of the common stopping criteria are a pure node, a given threshold for the minimum number of instances, and a given threshold for the minimum change in the class label's purity. Finally, a class variable is predicted in each leaf of the tree by either the average class variable in regression tree or the most frequent class variable in the classification tree. For classification problems, it is also possible to estimate the probabilities of each class variable and resembles the output of logistic regression.

2.2.1.2 Model-based Recursive Partitioning

Another form of recursive partitioning is the model-based recursive partitioning. The main idea of this algorithm is fitting a parametric model by computing a tree in which every leaf is associated with a fitted model (e.g. a linear regression). The model's objective function is used to estimate the parameters and the splitting points. The corresponding model scores are tested for parameter instability in each node to determine which variable should be picked for partitioning [Zeileis et al. 2008].

For example, modeling the dependency of a clinical response variable to the dosage of a medication can be estimated using linear regression. Experiment shows the model's coefficients are different for various age groups. Zeileis et al in [Zeileis et al. 2008] proposed a method to partition the training dataset to find instabilities of the regression's coefficients using a formula that measures the structural changes. If the algorithm finds an unstable variable like age, the variable will be divided into two or more partitions and make a tree.

Regression and classification trees have three main flaws:

- Instability: Decision trees are very sensitive to any change in the training dataset. Even a small change in the training dataset such as changing variables, removing duplicated instances can cause large changes in the tree.

- Non-optimality: The first step in forming a decision tree is choosing the best predictor variable as the root of the tree. If the root variable is not optimal, the decision tree does not perform well.

- Variable selection: Variable selection is usually biased in favor of the variables with certain characteristics. If there is a categorical variable with many distinct values or even a variable with many missing values, it has the higher chance to be chosen. Bias in variable selection

is carried forward to an ensemble of trees, specifically when the training dataset has predictor variables of different types such as numerical and categorical variables.

2.2.2 Ensemble Learning

One solution for overcoming the instability of regression trees and avoiding the overfitting problem is learning multiple trees (ensembles of trees) rather than a single tree. An ensemble method improves the performance by using multiple models instead of a single one. The class label is predicted by using the weighted vote of multiple ensemble models. Bagging and random forest are the two main ensemble learning algorithms. In both bagging [Breiman 1996] and random forest [Breiman 2001], multiple trees are built on random samples of the training dataset. In the first step, a sample of data points is selected randomly from the training dataset (with replacement) and then in the second step, a tree is grown on each random sample. In ensemble learning, predictions made by each tree need to be combined with other trees to make a final prediction. Averaging (weighted or unweighted) is one of the most common ways to do it.

2.2.2.1 Random Forest

Random forest [Breiman 2001] introduces another source of diversity in tree-based classifiers. While bagging and random forest are alike in using a diverse set of random samples, only random forest employs a modified tree learning algorithm that selects, a random subset of the predictor variables. Random forest algorithm has two parameters: number of the trees and number of the randomly selected predictor variables. Random forest can be a special case of bagging when the number of predictor variables in each iteration is fixed and is equal to the number of predictor variables in the training dataset. Having more diverse trees is the main reason that random forest models perform better than bagging.

One of the main issues with the ensemble learning algorithms is the interpretability. It is not easy to visualize an ensemble of trees. Each predictor variable may appear in different positions in the tree or even in different trees, and we cannot quantify variables positions to draw a unique graph as a visualization. Another issue with the random forest is the order effect. Order effect is important because only one variable is selected to split at a time. As a result, different orders can generate different trees and predictions.

Most of the regression and classification trees have the overfitting problem. Although Breiman in [Breiman 2001] claimed random forest does not overfit at all, Luellen et al. [Luellen et al. 2005], investigated the possibility of overfitting when the number of trees grows and Segal in [Segal 2004] argued that deeper trees have more risk of overfitting.

2.2.2.2 Boosting

Boosting is another type of ensemble learning techniques. The main idea of boosting is based on a question [Kearns and Valiant 1994]: "Can a set of weak learners create a set of strong learners?" Kearns and Valiant proved that weak learners (performing slightly better than random) can be boosted to form a stronger learner. Although Schapire in [Schapire 1990] provided a provable polynomial time boosting algorithm for the first time, the first practical boosting algorithm was AdaBoost (Adaptive Boosting) [Freund and Schapire 1997]. Adaboost develops an ensemble that incrementally adds one classifier at a time. In the beginning, each training instance will be weighted equally. Then a classifier is built on the sampled dataset and the weighted error of each data instances is calculated. If the weighted error is more than a pre-specified threshold, the weight will be increased, and it forces classifier to focus more on those incorrectly classified instances. Adaboost returns a set of k classifiers and updated weights of each data instance [Kuncheva 2004]. There is a misconception in the literature that boosting does not overfit even when the number of iterations is too large [Meir and Ratsch 2002]. Grove in [Adam J. Grove 1998] demonstrated a clear evidence of overfitting on some noisy datasets.

The main difference between ensemble learning methods such as random forest and boosting with CPXC is the universal classifier assumption; ensemble classifiers can be applied to all data points rather than just a subset of data points. Experiments show that CPXC often builds ensembles that are much smaller and much more accurate than those built by traditional ensemble approaches.

2.2.3 Pattern Based Classification

Another set of studies related to our research is pattern-based classification methods. Some of the well-know pattern based classifiers are [Dong et al. 1999],[Yin and Han 2003], [Li et al. 2004], and [Li et al. 2001] .According to [Zimmermann and Nijssen 2014], there are two approaches to mine patterns in order to classify an unseen test data point. In the first approach, class label is considered as an additional item in the dataset [Agrawal et al. 1996] and the second approach, the class label is considered as a separate dataset [Antonie and Zaiane 2002].

A common issue in most of the pattern mining problems is redundancy and contradictory patterns. In some cases, patterns are so similar or contradictory to each other and uses of methods to organize the patterns are required. Three categories of techniques are proposed to deal with this problem: (1) local evaluation, local modification [Li et al. 2001], (2) global evaluation, global modification [Bringmann and Zimmermann 2008], and (3) local evaluation, global modification [Zimmermann et al. 2010].

The last step of the pattern-based classification methods is predicting the class label of an unseen

data points. There are two groups of methods: direct classification and indirect classification. In direct classification methods, which is related to this study, patterns can be ordered according to some criterion and the first pattern that matches makes the prediction or there is a voting mechanism that collects all patterns that match the unseen data point and methods such as majority voting makes the final decision.

Contrast pattern Aided Classification (CPXC), one of the methodologies that we are proposing in this study, takes a different approach for pattern mining and making predictions. CPXC uses patterns as conditions and provides a local classifier for the matching dataset of a given pattern. In the final step, CPXC builds a patterns aided classifier (PXC) using a small set of pattern and local classifier pairs.

There are also some studies that follow the divide-and-conquer strategy. Typically, the input space is divided into nonoverlap clusters, followed which a local classifier is trained on instances falling in each cluster [Jacobs et al. 1991]. In [Dekel and Shamir 2012], they proposed a stochastic optimization algorithm to iteratively identify high error regions and learn specialized classifiers for those regions. Clustered SVM [Gu and Han 2013] is another algorithm that splits data into clusters and learns linear SVM models for each cluster. They also used some global regularization techniques to control overfitting.

2.3 Classification Methods on Imbalanced Datasets

There are two approaches to tackle classification problems on imbalanced datasets: cost sensitive learning methods and sampling techniques [Chen et al. 2004]. In classifying imbalanced datasets, the importance of minority (positive) instances is higher than that of majority (negative) instances. Therefore, the cost of misclassifying a minority instance is more than the cost of misclassifying a majority one. In cost sensitive learning methods, a cost matrix encodes the cost of classifying instances from one class as another. The goal of cost sensitive learning methods is to maximize the accuracy while minimizing the cost of misclassification [Pazzani et al. 1994]. Reported studies in cost sensitive learning methods fall into three main categories: weighting the data space [Zadrozny et al. 2003], making a learning algorithm cost sensitive [Ling et al. 2004], and using Bayes risk theory to assign each instance to its lowest risk class [Domingos 1999].

Sampling techniques may oversample the minority class, or undersample the majority class or, both. One of the first undersampling techniques is called SHRINK [Kubat and Matwin 1997]. SHRINK assigns labels to a mixed region as minority class regardless of whether the minority examples dominate in the region or not; then it searches for the best minority region. Ling and Li

[Ling and Li 1998] proposed an oversampling method to make the dataset more balanced. In their method, by replicating the minority instances, the weight of minority instances increases. SMOTE is another sampling method developed by Chawla et al. [Chawla and Bowyer 2002]. SMOTE combines oversampling and undersampling to improve classification performance comparing to just undersampling the majority class or oversampling the minority class. More than oversampling, SMOTE creates synthetic minority instances to boost the minority class.

2.4 Contrast patterns

Contrast pattern mining have received much attention recently [Dong and Bailey 2012]. Researchers have proposed numerous contrast pattern based methods for classification [Dong et al. 1999], clustering and clustering quality evaluation [Liu and Dong 2009], outlier detection [Chen and Dong 2006], bioinformatics [Mao and Dong 2005], cancer analysis [Li and Wong 2002], chemoinformatics [Auer and Bajorath 2006], and so on. This dissertation is novel in its focus on contrast pattern aided construction of accurate prediction models, and in the concept of using pattern and local regression model pairs to capture distinct predictor-response relationships involving multiple multidimensional interactions.

3

Preliminaries

In this section, we discuss about the following preliminaries required for CPXR and CPXC algorithms.

- Regression algorithms including linear and logistic regression
- Classification algorithms including naive Bayes and decision tree
- Discretization methods (Binning) including equi-width and entropy discretization
- Pattern mining

3.1 Numerical Prediction Modeling

Regression aims at modeling the effect of a given set of predictor variables $x_1, x_2, ..., x_k$, on a response variable y. The predictor variables are also called *independent*, *regressors*, or *explanatory* variables and the response variable y is also called *dependent* variable. The various modeling methods differ mainly through the type of dependent variable (categorical vs numerical) and types of predictor variables. In regression modeling, the relationship between the response and predictor variables is not a deterministic function and each function also has a random error. It assumes the response variable is a random variable and its distribution depends on the predictor variables. Therefore, we can only estimate the mean value of the response variable. In other words, we model the expected (conditional) value of y depending on predictor variables $x_1, x_2, ..., x_k$.

$$E(y|x_1, ..., x_k) = f(x_1, ..., x_k) + \epsilon \tag{3.1}$$

where ϵ represents the deviation from the expected value and called *residual error*. In the next two sections, we will discuss two popular regression methods.

3.1.1 Linear Regression

Linear regression, which is the most popular type of regression method, examines the relationship between a numerical response variable and a set of numerical or categorical predictor variables [Montgomery et al. 2015]. If there is a single predictor variable and a numerical response variable, it is called *simple linear regression*. Assuming k predictor variables $x_1, x_2, ..., x_k$, and a response variable y, the expected value of y is defined as a linear combination of predictor variables, i.e.,

$$E(y|x_1, ..., x_k) = f(x_1, ..., x_k) + \epsilon = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + \epsilon \tag{3.2}$$

where $\beta_1, ..., \beta_k$ are *coefficients* and need to be estimated and β_0 represents the *intercept*. One method to estimate the intercept and the coefficients in the linear regression is *least square* method. To measure the performance of a regression model, we use residual value. The *residual* of regression model f on an instance x_i is the difference between the predicted response variable value and observed response variable value, $f(x_i) - y_i$. Some other often used quality measures in linear regression models are the following:

- *Sum of Squared Error (SSE)* is the sum of squares of residuals. SSE of model f is given by

$$SSE(f) = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{3.3}$$

- *Root Mean Square Error (RMSE)* of model f is given by

$$RMSE(f) = \sqrt{\frac{\sum_{i=1}^{n} (y_i - f(x_i))^2}{n}} \tag{3.4}$$

- and R^2 *(R squared)* of model f is given by

$$R^2(f) = 1 - \frac{\sum_{i=1}^{n} (f(x_i) - y_i)^2}{\sum_{i=1}^{n} (y_i - avg_{j=1}^{n} y_j)^2} \tag{3.5}$$

  where y_i is the observed response variable value and $f(x_i)$ is the predicted response variable value of instance x_i under the linear regression model f.

3.1.2 Logistic Regression

The main limitation of linear regression is that it cannot deal with categorical response variables [Hosmer et al. 2013]. In many interesting applications in the world, we want to model a categorical or specifically a binary response variable. *Logistic regression* is an adaption of linear regression to handle categorical response variables. Logistic regression is very popular in different fields specifically in clinical prediction modeling. To show how much logistic regression is used over the years, we ran

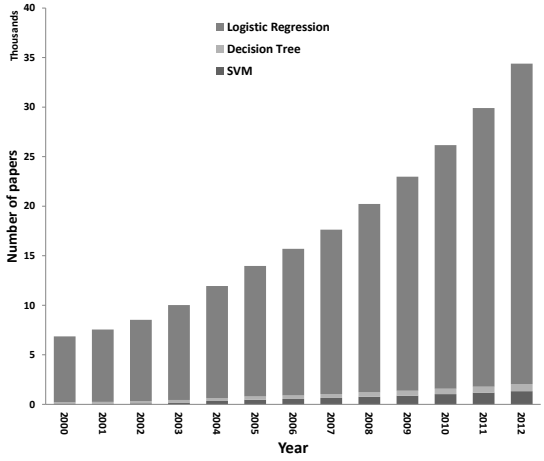


Figure 3.1: Number of articles published in PubMed with "Logistic Regression", "Decision Tree" and "Support Vector Machine" in the titles between 2000 and 2012.

a simple query on PubMed [National Center for Biotechnology and Medicine 2011] to extract the number of papers published in PubMed between 2002 and 2012 with "Logistic Regression", "Decision Tree" and "Support Vector Machine" in the titles. Figure 3.1 represents how much logistic regression has been used in medicine compared to decision tree and support vector machine.

Since the response variable is categorical, the linear regression's least square is not applicable anymore. Instead of conditional expectation in linear regression, conditional probability is being used in logistic regression. In order to modeling conditional probability $P(Y = 1|X = x)$ as a function of x, any unknown parameters in the function needs to be estimated by maximum likelihood method. Since probability (p) must be between 0 and 1, we have to use logarithmic transformation to unbound the probability (p). The next problem is that logarithms are bounded in one direction but linear functions are not. The easiest modification is the *logit transformation*, $log\frac{p}{1-p}$ [Hosmer and Lemeshow 2000]:

$$logit(p) = log\frac{p}{1-p} \tag{3.6}$$

and the formal equation of logistic regression is:

$$logit(p(X)) = log\frac{p(X)}{1-p(X)} = \beta_0 + \beta_0 + \beta_1 x_1 + ... + \beta_k x_k \tag{3.7}$$

and solving for $p(X)$:

$$p(X) = \frac{e^{\beta_0+\beta_0+\beta_1 x_1+...+\beta_k x_k}}{1+e^{\beta_0+\beta_0+\beta_1 x_1+...+\beta_k x_k}} = \frac{1}{1+e^{-(\beta_1 x_1+...+\beta_k x_k)}} \tag{3.8}$$

where $p(X)$ is the probability of the outcome of interest, e is natural logarithm, β_0 is the intercept

and β_i are the regression coefficients. Maximum likelihood will be used to determine regression coefficients.

As we discussed before, linear regression's residual is equal to the difference between predicted values and observed values. The same concept applies to logistic regression where the observed value is 0 or 1, and the predicted value is the probability (a number between 0 and 1). Standardized residual (Pearson residual) of a logistic regression model f on an instance X_i is

$$r_i = \frac{Y_i - p(X_i)}{\sqrt{p(X_i)(1 - p(X_i))}} \tag{3.9}$$

and then *Chi-square* (χ^2) of model f is

$$\chi^2 = \sum_{i=1}^{n} r_i^2 \tag{3.10}$$

Confusion matrix and ROC curves are other methods used to measure the performance of logistic regression models.

Odds Ratio: Odds ratio of each predictor variable is equal to the exponent of its coefficient. For example, odds ratio of x_i is e^{β_i}, where β_i is the coefficient of x_i, or inversely, logarithm of odds ratio is the coefficient. Odds ratio shows us how the probability of "success" changes with a one-unit change in the predictor variable. Increasing the odds of success means increasing the probability, and vice-versa. Therefore, the sign of the coefficient indicates the direction of its relationship: + means a positive relationship between the predictor variable and the likelihood of success, and - means a negative relationship. However, unlike the coefficients, all odds ratios are positive values. The distinction regarding a positive or negative relationship in the odds ratios is given by which side of 1 they fall on. 1 indicates no relationship, less than one indicates a negative relationship, and greater than one indicates a positive relationship.

3.2 Classification

3.2.1 Naive Bayes Classifier

Naive Bayes Classifier (NBC) is a very fast and simple classifier [Han et al. 2011]. NBC is based on Bayes theorem and can predict the class membership probabilities, such as the probability that a given instance belongs to a particular class. NBC is considered "naive", because it uses class conditional independence assumption. The class conditional independence assumption says the effect of a predictor variable value on a given class is independent of the values of the other predictor variables.

Given a dataset D including a set of n predictor variables $A_1, A_2, ..., A_n$, the probability model

for a classifier is

$$P(C \mid A_1, A_2, ..., A_n) \tag{3.11}$$

over a response class variable C, conditional on a set of variables $A_1, A_2, ..., A_n$. In other words, (3.11) calculates the probability of class C considering the values of $A_1, A_2, ..., A_n$. We reformulate (3.11) using Bayes theorem

$$P(C \mid A_1, A_2, ..., A_n) = \frac{P(C)P(A_1, A_2, ..., A_n \mid C)}{P(A_1, A_2, ..., A_n)} \tag{3.12}$$

Since the value of denominator does not depend to C, its calculation is straightforward and is effectively constant. The nominator can also be written as follows (applying conditional probability definition)

$$\begin{aligned} &= P(C)P(A_1, A_2, ..., A_n \mid C) \\ &= P(C)P(A_1 \mid C)P(A_2, ..., A_n \mid C, A_1) \\ &... \\ &= P(C)P(A_1 \mid C)P(A_2 \mid C, A_1)P(A_3 \mid C, A_1, A_2)...P(A_n \mid C, A_1, A_2, ..., A_{n-1}) \end{aligned} \tag{3.13}$$

and now we use the class conditional independence assumption. Since every variable A_i is independent from A_j for $i \neq j$, then

$$P(A_i \mid C, A_j) = P(A_i \mid C) \tag{3.14}$$

and (3.13) can be expressed as

$$= P(C)P(A_1 \mid C)P(A_2 \mid C)...P(A_n \mid C) = P(C)\prod_{i=1}^{n} P(A_i \mid C) \tag{3.15}$$

and this means (3.11) can be calculated as follows

$$P(C \mid A_1, A_2, ..., A_n) = \frac{1}{Z}P(C)\prod_{i=1}^{n} P(A_i \mid C) \tag{3.16}$$

where Z is a constant when the values of predictor variables are known.

3.2.2 Decision Tree

Decision trees are powerful and popular tools for classification. A decision tree is in the form of a tree structure, where each node is either a leaf node or a decision node. A leaf node specifies the value of the class variable. To determine the class variable's value of an unseen example, we need to start from the root and move through it based on the value of the predictor variables to a leaf node, which provides the class variable's value of the instance.

C4.5 is one of the algorithms used to construct a decision tree [Quinlan 1993]. Given a dataset D, the following two steps are used to initialize C4.5 trees:

- If the class variable of all instances in D are the same, then tree has just one leaf level.
- Otherwise, choose a single variable with at least two distinct values. Make this variable the root of your tree. Create a branch for each distinct value and split D into D_1, D_2, ... based on the value of the root's variable. Apply the same procedure recursively to each subset.

There are many ways to choose the root variable and the splitting points (for numerical variables). The root variable should be chosen in a way that effectively splits instances into subsets enriched in one class or the other. The normalized information gain is one of the methods to determine the splitting criterion. The variable with the highest normalized information gain can be chosen. There are also some pruning methods to avoid overfitting. Pruning is carried out from leaves to the root [Kearns and Mansour 1998].

3.3 Discretization Methods (Binning)

In both CPXR and CPXC algorithms, we need to transform numerical predictor variables into categorical counterparts (bins). *Equi-width* and *Entropy* are two discretization methods that were used in our algorithms.

3.3.1 Equi-width Discretization

Equi-width divides each predictor variable A into k intervals of equal sizes. The width of intervals is defined by:

$$w = \frac{max(A) - min(A)}{k} \tag{3.17}$$

k is determined by the user and the interval boundaries are the followings:

$$[min(A), min(A) + w), [min(A) + w, min(A) + 2w), ..., [min(A) + (k-1)w, max(A)] \tag{3.18}$$

Example: Let's assume $A = \{0, 4, 12, 16, 16, 18, 24, 24\}$ is a numerical variable and $k = 3$. Equi-width intervals are $[0, 8), [8, 16), [16, 24]$ and then
$0, 4 \in Bin_1 = [0, 8)$
$12 \in Bin_2 = [8, 16)$
$16, 16, 18, 24, 24 \in Bin_3 = [16, 24]$
and $A_{disc} = \{Bin_1, Bin_1, Bin_2, Bin_3, Bin_3, Bin_3, Bin_3, Bin_3\}$

3.3.2 Entropy Discretization

Entropy discretization (binning) method is an iterative splitting based approach uses the class variable to find the best splits so that the bins are as pure as possible, that is the majority of the

values in a bin have the same class label. Formally, it is characterized by finding the splits with the maximal information gain. Assuming A_i is a numerical variable with a binary class label, entropy binning steps are the followings:

1. Calculate *Entropy* for the class label using:

$$E(A) = -p_1 log_2^{p_1} - p_0 log_2^{p_0} \tag{3.19}$$

where $p_1 = \frac{|c_1|}{|A|}$ and $p_0 = \frac{|c_0|}{|A|}$, $|C_1|$ and $|C_0|$ are the number of instances with class labels 1 and 0 respectively, and $|A|$ is the cardinality of A.

2. Given a split point v, calculate entropy for the class label using:

$$E(A, v) = \frac{|S_{<v}|}{S} E(S_{<v}) + \frac{|S_{\geq v}|}{S} E(S_{\geq v}) \tag{3.20}$$

where $|S_{<v}|$ is the number of instances of A in $[min(A), v)$ and $|S_{\geq v}|$ is the number of instances of A in $[v, max(A)]$.

3. Given a split point v, calculate *Information Gain (IG)* using:

$$IG(A, v) = E(A) - E(A, v) \tag{3.21}$$

4. Repeat steps 2 and 3 to find split point with the maximum information gain. The new intervals may be split further until a termination criterion [Fayyad and Irani] is met.

Example: To illustrate entropy binning method, we consider example data in table 3.1.

Table 3.1: A numerical variable A with a binary class label C

TID	A	C	TID	A	C	TID	A	C
1	53	1	9	68	0	17	75	0
2	56	1	10	69	0	18	75	1
3	57	1	11	70	0	19	76	0
4	63	0	12	70	1	20	76	0
5	66	0	13	70	1	21	78	0
6	67	0	14	70	1	22	79	0
7	67	0	15	72	0	23	80	0
8	67	0	16	73	0	24	81	0

1. Calculate entropy of A for the class label C:
   $E(A) = -\frac{7}{24}log_2^{(\frac{7}{24})} - \frac{17}{24}log_2^{(\frac{17}{24})} = -0.29log_2^{0.29} - 0.71log_2^{0.71} = 0.871$

2. Given a split point like $v = 60$, calculate entropy of A for the class label C:
   $E(A, 60) = \frac{|S_{<60}|}{S}E(S_{<60}) + \frac{|S_{\geq 60}|}{S}E(S_{\geq 60}) = \frac{3}{24}E(S_{<60}) + \frac{21}{24}E(S_{\geq 60}) = \frac{3}{24} \times 0 + \frac{21}{24} \times 0.7 = 0.516$
   such that
   $E(S_{<60}) = \frac{3}{3}log_2^1 - \frac{0}{3}log_2^0 = 0$
   $E(S_{\geq 60}) = \frac{4}{21}log_2^{\frac{4}{21}} - \frac{17}{21}log_2^{\frac{17}{21}} = 0.7$

3. Given a split point $v = 60$, calculate information gain IG:
   $IG(A, 60) = E(A) - E(A, 60) = 0.871 - 0.615 = 0.256$

4. Repeat steps 2 and 3 to find split point with maximal information gain.
   $IG(A, 60) = \mathbf{0.256}$
   $IG(A, 70) = 0.101$
   $IG(A, 75) = 0.148$
   The information gain for all three split points (bins) show that $v = 60$ is the most informative split point and returns the highest gain.

3.4 Contrast Pattern Mining

The next step, after numerical predictor variables are discretized, is mining the contrast patterns. In this section, we define the concepts related to contrast pattern mining. Let $D = \{(X_i, Y_i) \mid 1 \leq i \leq n\}$ be a training dataset including pairs of predictor variables values and response variable values. Each X_i is a vector of k predictor variables, $x_1, ..., x_k$.

Definition: An *item* is a *conditioned variable* of the form "$x_i = a$" if x_i is categorical, or "$a \leq x_i < b$" if x_i is numerical. For example, assuming a dataset D with 6 predictor variables and a binary response (class) variable, Table 3.2 can be used to illustrate concepts defined in this section.

Dataset D has 3 numerical (x_1, x_2, x_3) and 3 categorical variables (x_4, x_5, x_6). We used equi-width binning method to discretize numerical variables. Table 3.3 represents the discretized version of dataset D (Table 3.2). Each cell of Table 3.3 is in the d_{ij} format such that i is the index of variable and j is the index of bin. For example, $d_{12} \in t_1$ means the value of x_1 in t_1 belongs to the second bin ($x_1 \in [1.25, 2, 5)$). Since x_3, x_4 and x_5 are already categorical, we just convert those to the new format. For example, $d_{41} \in t_5$ means the value of x_4 in t_5 is in the first category (x_4 has 4 possible values.)

Each element of Table 3.3 is an item. For example, d_{12} is an item in the form of "$x_1 \in [1.25, 2.5)$"

Table 3.2: A toy dataset

TID	x_1	x_2	x_3	x_4	x_5	x_6	**Class**
t_1	1.7	23	0.3	1	1	2	1
t_2	3.02	99	0.4	2	1	1	1
t_3	0	34	0.2	3	1	3	0
t_4	0.13	32	0.9	4	0	2	1
t_5	2.4	77	0.8	1	1	2	0
t_6	5.01	90	0.1	2	0	3	0
t_7	0.45	65	0.1	3	0	1	0
t_8	1.43	7	0.2	4	0	2	0
t_9	4.1	11	0.3	1	0	2	1
t_{10}	0.99	43	0.5	2	1	3	0

or "$1.25 \leq x_1 < 2.5$" and d_{41} is another item in the form of "$x_4 = 1$".

Definition: A *pattern* or an *itemset* is a finite set of items.

For example, $p_1 = \{d_{11}\}$ is a pattern with one item and can be represented in the form of $\{x_1 < 1.25\}$ and $p_2 = \{d_{11}, d_{31}, d_{43}\}$ is another pattern with 3 items and can be represented in the form of $\{x_1 < 1.25$ AND $x_3 < 0.3$ AND $x_4 = 3\}$.

Definition: An instance t is said to *match* a pattern P, denoted by $t \models P$, if t satisfies every item in P or in another word $P \subseteq t$. For example, t_3 matches p_2 because t_3 satisfies every item in p_2. The *matching dataset* of pattern P in dataset D is $mds(P, D) = \{x \in D \mid x \models P\}$. For example, $mds(p_1, D) = \{t_3, t_4, t_7, t_8, t_{10}\}$ and $mds(p_2, D) = \{t_3, t_7\}$.

Definition: The *support* of pattern P in dataset D is $supp(P, D) = \frac{|mds(P,D)|}{|D|}$ and then $supp(p_1, D) = \frac{5}{10} = 0.5$. The support of a pattern can also be computed in a subset of a dataset. For example, $supp(P, C_1) = \frac{|mds(P,C_1)|}{|C_1|}$ where C_1 is a subset of dataset D which their class label is 1.

Definition: Given a support threshold s, a pattern P is called *frequent pattern* (also called *frequent itemset*) if $supp(P, D) \geq s$.

Definition: Given two data classes C_1 and C_2 (e.g., instances which their class variable values are 1 and 0 respectively.), the *support ratio* of pattern P (also called *growth ratio*) from C_1 to C_2 is $supp\_ratio_{C_1}^{C_2} = \frac{supp(P,C_1)}{supp(P,C_2)}$.

The next part of preliminaries are related to the concept of contrast patterns. A pattern is contrast pattern if its support is different within different classes.

Definition: [Dong and Li 1999] Given a support ratio threshold γ, a *contrast pattern* (also called

Table 3.3: Discretized version of dataset D in Table 3.2

TID	x_1	x_2	x_3	x_4	x_5	x_6	**Class**
t_1	d_{12}	d_{21}	d_{32}	d_{41}	d_{52}	d_{62}	1
t_2	d_{13}	d_{24}	d_{32}	d_{42}	d_{52}	d_{61}	1
t_3	d_{11}	d_{22}	d_{31}	d_{43}	d_{52}	d_{63}	0
t_4	d_{11}	d_{22}	d_{34}	d_{44}	d_{51}	d_{62}	1
t_5	d_{12}	d_{24}	d_{34}	d_{41}	d_{52}	d_{62}	0
t_6	d_{14}	d_{24}	d_{31}	d_{42}	d_{51}	d_{63}	0
t_7	d_{11}	d_{23}	d_{31}	d_{43}	d_{51}	d_{61}	0
t_8	d_{11}	d_{21}	d_{31}	d_{44}	d_{51}	d_{62}	0
t_9	d_{14}	d_{21}	d_{32}	d_{41}	d_{51}	d_{62}	1
t_{10}	d_{11}	d_{22}	d_{33}	d_{42}	d_{52}	d_{63}	0

emerging pattern) of class C_2 is a pattern P satisfying $supp\_ratio_{C_1}^{C_2} \geq \gamma$. Assuming $\gamma = 0.25$, $p_1 = \{d_{11}\}$ is a contrast pattern with the $supp\_ratio_{C_1}^{C_2} = \frac{1}{4} = 0.25$. $p_3 = \{d_{31}, d_{51}\}$ is another contrast pattern (called *jumping pattern*). p_3 is frequent in C_2 ($supp(p_3, C_2) = 3$) but non-frequent in C_1 ($supp(p_3, C_1) = 0$).

Equivalence classes is another important concept needs to be defined.

Definition: An *equivalence class (EC)* is a set of patterns with the same matching datasets, i.e., $EC(P) = \{Qs \mid mds(Q) = mds(P)\}$ where Qs are also patterns [Li et al. 2007]. The minimal patterns are called minimal generators (MG) and the longest ones are called closed patterns. For example, given dataset $D = \{abcdeghi, acdg, bcdghi, abdhi, bceghi\}$ of five transactions, one of the equivalence classes is *bcdghi* and it contains six MGs: $bcd, bdg, cdh, cdi, dgh, dgi$. Each of those MGs presents a unique minimal combinations of single-item conditions that differentiates this EC from other ECs.

4

Contrast Pattern Aided Regression (CPXR)

In chapter 3, we discussed about all preliminaries required for CPXR. In this chapter, we first introduce the main idea of this research in the form of a simple example. We then present the concept of Pattern Aided Regression (PXR) and Contrast Pattern Aided Regression (CPXR) algorithm. We also evaluate CPXR methodology against a set of benchmark and synthetic datasets.

4.1 PXR Concept

In order to introduce the PXR concept, which is the main idea of CPXR methodology, following definitions need to be introduced:

Definition: Given a training dataset $D = \{(x_i, y_i) \mid 1 \leq i \leq n\}$, a regression model built on D is called *baseline model* and shown by f_0.

In this research, we used linear regression and piecewise regression to build baseline models. However, any numerical prediction baseline modeling method would work perfectly.

Definition: Given the matching dataset of pattern P, $mds(P, D)$, a regression model built on $mds(P, D)$ is called *local model* and shown by f_P.

f_Ps use all predictor variables used in building baseline model. Similar to baseline models, any numerical prediction methods can be used in order to build the local models.

As discussed earlier, the predictor-response (PR) relationship is one of the key points of the CPXR methodology. The most convenient way to represent predictor-response relationships is pairing patterns and local models. Each pattern represents a region in the data space. The local model associated with the pattern represents how the response variable depends on the predictor variables

for the data instances that matches the pattern. Each pair (P, f_P) represents a different PR relationship and should be highly different from other PR relationships. A small set of (P, f_P) pairs can represent a diverse set of PR relationships.

Definition: A *pattern aided regression (PXR)* is a tuple $PXR = ((P_1, f_1, w_1), ..., (P_k, f_k, w_k), f_d)$, where k is a positive integer, $P_1, ..., P_k$ are patterns, $f_1, ..., f_k, f_d$ are regression models, and $w_1, ..., w_k > 0$ are weights. $\{P_1, ..., P_k\}$ is the pattern set, $\{f_1, ..., f_k\}$ is the set of local models associated to the patterns in the pattern set and f_d is the default model. We define the regression function of PXR as

$$f_{PXR}(x) = \begin{cases} \frac{\Sigma_{P_i \in \pi_x} w_i f_i(x)}{\Sigma_{P_i \in \pi_x} w_i} & \text{if } \pi_x \neq \emptyset \\ f_d(x) & \text{otherwise} \end{cases} \tag{4.1}$$

for instance x,

where $\pi_x = \{P_i \mid 1 \leq i \leq k, x \text{ satisfies } P_i\}$.

If x is matched by just one pattern like P_i, then $f_{PXR}(x)$ is equal to $f_i(x)$, the local model associated to pattern P_i. If x is matched by more than one pattern, the weighted average of the predicted values of local models is used to determine $f_{PXR}(x)$. If x does not match by any of the patterns in the pattern set, the default model $f_d(x)$, will be used to determine $f_{PXR}(x)$. As will be discussed later, the weight of each local model can be determined based on the effects of the local model on the residual reduction.

4.2 CPXR Example

To illustrate PXR concept, we present a very simple example. Table 4.1 shows the data points values for a given dataset D with one predictor variable X and one response variable Y. Dataset D has 30 data points (instances), and both predictor and response variables are numerical[1].

Figure 4.1 is the graphical representation of dataset D on a 2D plane along with a series of fitted models. Looking at figure 4.1, we observe that data points are not uniformly distributed. The synthetic dataset D, like many real multi-modal datasets that we experimented on, is extremely heterogeneous.

In order to show the differences between PXR model and traditional regression models, we applied simple linear regression and piecewise linear regression (PLR) on the same dataset. Black dashed line is representing the fitted linear regression model and black solid line is the fitted model by piecewise regression. Colored dashed lines are the PXR model and the mathematical form of

[1] In this example, we used just one predictor variable in order to have a graphical representation. CPXR methodology can be applied to datasets with any number of predictor variables.

Table 4.1: A toy dataset to explain a PXR model

TID	X	Y	TID	X	Y
1	1.1	4.6	16	3.01	1.7
2	1.2	4.6	17	3.56	1.6
3	1.23	4.2	18	3.69	1.5
4	1.34	4.9	19	3.21	1.9
5	1.34	4.03	20	3.3	1.2
6	1.32	4.78	21	5.2	3.9
7	1.29	4.21	22	5.25	3.1
8	1.26	4.78	23	5.43	3.8
9	1.19	4.5	24	5.55	3.7
10	1.2	4.6	25	5.65	3.6
11	3.8	1.7	26	5.7	3.6
12	3.1	1.1	27	5.73	3.5
13	3.4	1.9	28	5.9	3.7
14	3.45	1.2	29	5.95	3.2
15	3.67	1.8	30	5.95	3.9

PXR model is:

$$PXR = \begin{cases} 4.83 - 0.25x & \text{if } x \leq 2 \\ 0.76 + 0.23x & \text{if } 3 < x \leq 4 \\ 3.62 & \text{if } x > 5 \\ 3.92 - 0.2x & \text{Otherwise} \end{cases} \tag{4.2}$$

Equation 4.2, which is a multi-valued model, contains a PXR model includes a set of linear regression models and a set of conditions (patterns), and each model is just applicable on the region of data space defined by the pattern. For example, if the value of predictor variable x is between 3 and 4, then the second equation is used to estimate the value of response variable y. For the values of x more than 5, the response variable is constant and equal to 3.62. Finally, if the value of x does not fall in any of those conditions, we should use the last model that is same as simple linear regression (dashed line).

Table 4.2 compares the accuracy of CPXR with other regression methods such as piecewise and linear regression, and it is obvious that CPXR is more accurate than other methods in terms of RMSE and R^2. The second observation is that PXR's local models are diverse and different from

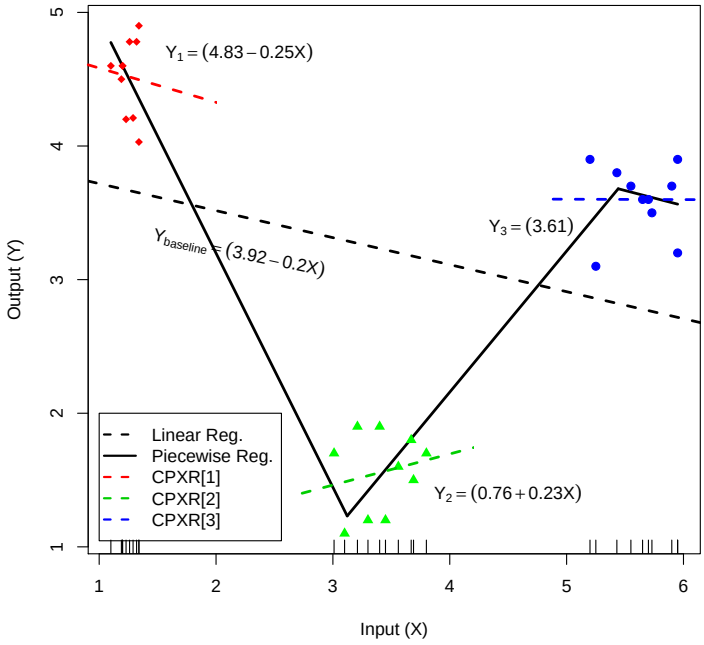


Figure 4.1: A graphical representation of dataset D

each other. For example, the slope of x in the first model is negative, in the second model is positive and in the third model is zero which shows how much different subgroups of data points behave differently and should be treated in separate models, not a global model. As mentioned before, CPXR returns a PXR model including a set of local models and patterns that each local model is associated with a pattern.

Table 4.2: Accuracy of different regression methods on dataset D

Regression method	**RMSE**	R^2
Linear regression	1.21	0.08
Piecewise regression	0.29	0.945
CPXR	**0.26**	**0.955**

4.3 CPXR Algorithm

In this section, we describe the CPXR algorithm along two quality measures that are used by the algorithm.

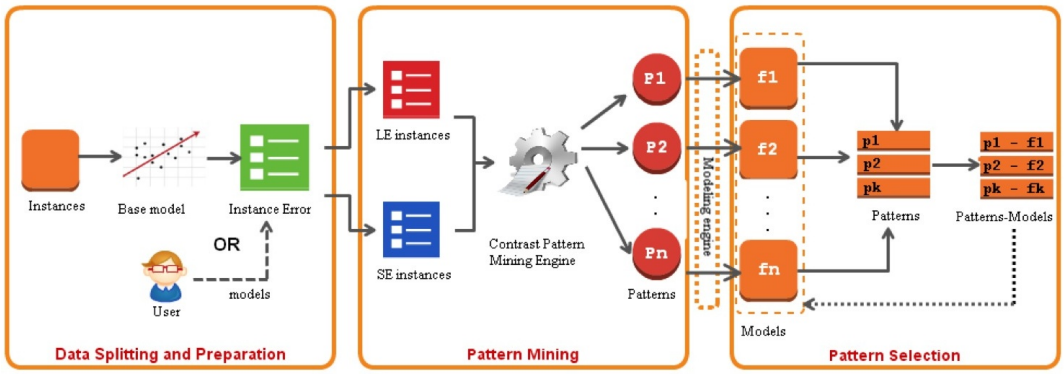


Figure 4.2: A graphical representation of CPXR algorithm

4.3.1 Quality Measures on Patterns and Pattern Sets

The core of CPXR methodology is finding a set of patterns along with a set of local models that correct the prediction errors of the baseline model. To do this, we first need a mechanism to measure how much a pattern and a PXR (see definition 4.1) can reduce the residual errors.

The following definition represents a measure to quantify the residual reduction of a PXR.

Definition: The *total residual reduction* (trr) of a PXR, $((P_1, f_1, w_1), ..., (P_k, f_k, w_k), f_d)$ is

$$\mathsf{trr}(PXR) = \frac{\Sigma_{x \in \mathsf{mds}(PS)} |r_x(f)| - \Sigma_{x \in \mathsf{mds}(PS)} |r_x(f_{PXR})|}{\Sigma_{x \in D} |r_x(f)|} \tag{4.3}$$

where $PS = \{P_1, ..., P_k\}$ is the pattern set, $r_x(f)$ is the f's residual on an instance x, $r_x(f_{PXR})$ is PXR's residual on an instance x and $mds(PS) = \cup_{P \in PS}\, mds(P)$.

So trr(PXR) measures the total residual reduction achieved by f_{PXR} as a fraction of f's total residual.

The huge number of contrast patterns is one of the issues in any pattern mining problems, and we need a mechanism to reduce the number of patterns and save computation time. Our experiments on real datasets revealed that some of the patterns have low utility, i.e. cannot reduce residual error of the baseline model. We need a method to measure the quality of individual patterns for the purpose of removing low utility patterns.

Definition: The *average residual reduction* (arr) of a pattern P w.r.t. a prediction model f and a dataset D is

$$\mathsf{arr}(P) = \frac{\Sigma_{x \in \mathsf{mds}(P)} |r_x(f)| - \Sigma_{x \in \mathsf{mds}(P)} |r_x(f_P)|}{|\mathsf{mds}(P)|} \tag{4.4}$$

where $r_x(f_P)$ is the f_P's residual on an instance x.

More than a filtering method, *arr* is also used as a weighting method (w_i in definition 4.1) for patterns in CPXR prediction models.

4.3.2 Algorithm

CPXR algorithm has three phases: data splitting and preparation, pattern mining, and pattern selection. In data splitting and preparation phase, CPXR splits the dataset D into two parts called LE and SE. The second phase, pattern mining, mines the contrast patterns on LE and removes some of the patterns after applying a set of filters. In the last phase, pattern selection, a double loop is used to search for a desirable pattern set in order to maximize trr: The inner loop performs pattern replacement, and the outer loop adds a new pattern to the pattern set and calls the inner loop at each iteration. The loops stop when they meet the termination criterion. Figure 4.2 is a diagram representing main steps of CPXR algorithm.

Let's describe different steps of algorithm 1. CPXR takes four inputs: a training dataset D, a baseline model f, a ratio ρ for splitting D into LE and SE, and a $minSup$ threshold on contrast patterns. You can substitute the baseline model f with the predicted values of data instances or the residual values of data instances corresponded to f.

In the first step, CPXR splits dataset D into LE and SE using a splitting point. To determine the splitting point, we use the magnitude of residual error as

$$\rho \approx \frac{\sum_{x_i \in LE} |r_i|}{\sum_{x_i \in D} |r_i|} \tag{4.5}$$

to find the index of sorted splitting instance (Steps 1.2, 1.3). In step 1.4, data instances with the index more than κ are located in LE and other instances are located in SE. Step 1.5 uses entropy based binning and equi-width binning methods to discretize numerical variables into the disjoint intervals. If entropy binning method cannot find the split points, we use equi-width to discretize the numerical variables. If all variables are categorical, CPXR skips 1.5. Contrast patterns in LE are mined in step 1.6. We used GcGrowth [Li et al. 2005] to mine contrast patterns. From each equivalence class, the shortest pattern is chosen and added to the contrast pattern set (CPS). In the next step, the matching dataset of each pattern $P \in CPS$ is constructed, and a local regression model is built for $mds(P)$. CPXR selects P_0 with the highest arr to initialize PS (Step 1.8). In steps 1.10 to 1.14, CPXR adds one pattern to PS in each iteration and improves the performance of PS gradually. If RMSE reduction in each iteration is less than 1%, the process is terminated. Some of the data instances are not covered by any of the patterns in PS. CPXR trains default model f_d, on those instances not covered in step 1.15. In the last step, CPXR returns PXR as a set of patterns, local models, weights and default model.

In algorithm 1, CPXR calls *IteratImp(CPS,PS)* to iteratively improve the performance of PS. The core of algorithm 2 is finding the best replacement for a pattern and locate it in PS. The

Input: (1) training data $D = \{(x_i, y_i)|1 \leq i \leq n\}$

(2) the baseline model f or predicted value of x_i under f or residual value of x_i under f

(3) a number ρ to partition D into LE, SE

(4) a *minSup* threshold on contrast patterns

Output: A PXR model

Let $r_1, ..., r_n$ denote f's residuals on $x_1, ..., x_n$;

Sort $x_1, ..., x_n$ based on residuals $r_1, ..., r_n$ ascendingly;

Determine κ to minimize $|\rho - \frac{\sum_{r_i > \kappa} |r_i|}{\sum |r_i|}|$;

Let $LE = \{x_i \mid r_i > \kappa\}$, $SE = D - LE$;

Discretize each numerical variable using a discretization method w.r.t. the LE and SE classes;

Extract all contrast patterns for *minSup* in the LE class, and select just one pattern having shortest length from each equivalence class for inclusion in the CPS set of contrast patterns, and apply other filters to remove patterns of small residual improvement;

For each $P \in CPS$, build the local regression model f_P for data in $\mathsf{mds}(P)$;

Let $PS = \{P_0\}$, where P_0 is the pattern P in CPS with highest arr;

repeat

  // add a pattern to PS

  Let P be the pattern in $CPS - PS$ that maximizes $\Delta(f_{PXR(PS \cup \{P\}, f)}, f)$;

  // $\Delta(f', f)$ denotes $\frac{RMSE(f) - RMSE(f')}{RMSE(f)}$

  Let $PS^o = PS$;

  if $\Delta(f_{PXR(PS \cup \{P\}, f)}, f) > 0$ **then**

    Let $PS = PS^o \cup \{P\}$;

    Call *IteratImp(CPS,PS)* to improve PS;

  end

until $\Delta(f_{PXR(PS,f)}, f) - \Delta(f_{PXR(PS^o,f)}, f) < 0.01$;

Let f_d be the regression model trained from $D - \cup_{P \in PS} \mathsf{mds}(P)$;

Return PXR;

Algorithm 1: The CPXR Algorithm

2.1 Let $impval = 1$;
2.2 **repeat**
2.3 | **for** *each* $P \in PS$ **do**
2.4 | | Let Q_P be a pattern P' in $CPS - PS$ maximizing $imp(PS, P, P')$;
| | **end**
2.5 | Let P_- be a pattern $P \in PS$ maximizing $imp(PS, P, Q_P)$;
2.6 | Let $impval = imp(PS, P_-, Q_{P_-})$;
2.7 | **if** $impval > 0$ **then** $PS = PS - \{P_-\} \cup \{Q_{P_-}\}$;
until $impval < 0.001$;

Algorithm 2: The $IteratImp(CPS, PS)$ Function

improvement of replacing P_- in PS by $P_+ \in CPS - PS$ is measured by:

$$imp(PS, P_-, P_+) = \mathsf{trr}(PXR') - \mathsf{trr}(PXR) \tag{4.6}$$

where PS in PXR' is equal to $PS - \{P_-\} \cup \{P_+\}$.

4.3.3 Techniques to Improve Computational Efficiency

Step 1.6 of algorithm 1 returns a huge number of patterns. In order to eliminate useless patterns without losing any desirable pattern, the following filters are applied:

- We remove contrast patterns of LE with the support ratio less than 1. If the support ratio is less than 1, there is no contrast between LE and SE classes.
- We remove patterns with small average residual reduction of f, i.e., $\Sigma_{x\in\mathsf{mds}(P)}|r_x(f)|-\Sigma_{x\in\mathsf{mds}(P)}|r_x(f_P)| \leq 0.01\Sigma_{x\in\mathsf{mds}(P)}|r_x(f)|$.
- The matching dataset of some of the patterns in CPS are similar to each other. We use Jaccard similarity to measure the similarity between the matching datasets of each pair of patterns. The Jaccard similarity between P_1 and P_2 is defined as:

  $$J(P_1, P_2) = \frac{|mds(P_1) \cap mds(P_2)|}{|mds(P_1) \cup mds(P_2)|} \tag{4.7}$$

  If the similarity is more than 0.9, then the pattern with less *arr* is removed. This filter is optional and it can be used if needed.
- If the number of elements in a pattern's matching dataset is less than the number of predictor variables, the pattern is removed. Another benefit of this filter is to control overfitting of the PXR models.

4.4 Experimental Results

In this section, we report a systematic evaluation of CPXR methodology. We used 50 real datasets and 23 synthetic datasets to evaluate the performance of CPXR in comparison with several well-known state-of-the-art regression methods. We evaluated CPXR from different aspects such as accuracy, sensitivity to noise, representability, and overfitting. The results show CPXR consistently outperforms competing methods often by big margins. We also discussed the impact of several parameters such as *minSup* on the CPXR performance. We examined the performance of CPXR when the baseline modeling method was not linear regression. The CPXR's computation time and memory usage is also investigated.

4.4.1 Datasets, Regression Methods and Parameters

We collected 50 real datasets from different sources. 43 datasets are from [Kim et al. 2007] and other datasets are from UCI repository [Bache and Lichman 2013] and other resources [Yeh 1998][Tsanas and Xifara 2012][Akbilgic et al. 2013][Hukkelhoven et al. 2005]. In table 4.3, each row represents some characteristics of the datasets including the number of instances and the number of predictor variables. For most of the datasets from [Kim et al. 2007], we used the splits given by [Chipman et al. 2012] that is 6-fold cross-validation with 20 repetitions.

We also generated 23 synthetic datasets that each one has different characteristics and used to evaluate the performance of CPXR. Each such dataset is generated based on a baseline regression model f, a set of patterns $PS = \{P_1, P_2, ..., P_k\}$ over the predictor variables, a local regression model f_i for each P_i, as well as applying 4 levels of noise, 0.05, 0.1, 0.15, 0.2. All predictor variable values were generated independently and identically distributed over [0, 1]. The response variables were generated to fit the regression model f_{PXR} with added noise (y is defined as the product of the given noise level, a random value in $[-1, 1]$, and the value given by f_{PXR}).

We used seven parameters to generate 23 synthetic datasets (listed in table 4.4): the numbers of instances, of variables, and of patterns; the condition overlap between the patterns (in terms of the number of shared common "single-variable" conditions), the magnitude of the coefficients, the level of difference between the largest coefficients in different regression models (indicated by the ratio of the largest coefficients), and the level of noise. We believe these synthetic datasets can cover all possible situations and present an honest judgment about the CPXR's performance.

We compared CPXR against several modeling techniques including: Linear Regression (LR), Piecewise Linear Regression (PLR)[McZgee and Carleton 1970], Support Vector Regression (SVR)[Smola and Vapnik 1997], Bayesian Additive Regression Trees (BART)[Chipman et al. 2012], and Gradient

Table 4.3: RMSE reduction of traditional methods and CPXR(LL,LP,LL:Regularized) over LR

Dataset	**#instances**	**#variables**	**PLR**	**SVR**	**BART**	**GBM**	**CPXR**(LL)	**CPXR**(LP)	**CPXR(LL:Regularized)**
Abalone	4177	8	4.5	0.00	3.18	1.36	12.33	**14.25**	18.67
Alcohol	2467	18	12.58	10.6	20.53	11.26	24.83	**26.14**	21.71
Amenity	3044	21	34.89	29.24	41.34	39.11	39.68	**42.19**	5.85
Attend	838	9	11.24	2.4	28.07	19.58	16.54	**30.43**	39.08
Baskball	96	4	21.93	11.23	8.02	4.81	53.66	**54.1**	40.91
Budget	1729	10	37.48	26.81	**91.58**	84.46	76.30	80.58	76.15
Cane	3775	9	8.51	0.00	20.15	16.42	23.93	**25.97**	21.48
Cardio	375	9	-18.42	-0.71	-0.21	-15.63	43.97	**49.09**	45.47
College	694	24	14.9	2.65	11.33	4.78	43.03	**46.73**	49.27
Concrete	1030	9	43.76	19.41	27.47	-48.18	41.36	**48.17**	50.48
County	3114	13	11.94	3.67	26.29	23.42	**32.33**	29.18	31.42
CPS	534	10	-3.75	-10.00	-5.00	-5.00	4.65	**4.92**	27.82
CPS95	21252	14	10.81	16.5	55.29	21.46	**65.57**	59.69	55.74
CPU	209	7	33.31	10.54	41.51	-59.65	57.78	**58.09**	63.04
Deer	654	13	-29.14	-28.95	-21.00	-28.34	50.07	**52.04**	47.00
Diabetes	375	15	7.43	6.43	4.42	4.71	32.32	**34.11**	46.85
Edu	1400	5	10.78	5.75	10.66	10.18	11.13	**13.45**	18.08
Energy Eff.	768	8	64.65	18.52	42.42	**80.07**	59.47	61.14	58.68
Engel	11986	5	6.72	2.99	5.37	5.22	11.2	**13.93**	18.79
Enroll	258	6	10.6	-6.79	0.66	-7.62	21.6	**22.73**	41.94
Fame	1319	22	43.95	11.58	41.91	24.36	55.11	**58.1**	43.43
Fat	252	14	16.04	4.4	2.86	-0.88	**45.88**	44.89	49.65
Fishery	6806	14	1.34	15.53	**35.73**	33.05	13.51	19.09	34.58
Houses	6880	8	6.93	1.35	25.13	24.34	23.49	**39.48**	23.61
Insur	2182	6	29.82	-0.26	33.68	**83.65**	40.49	39.07	39.95
Istanbul SE	536	8	23.73	5.08	22.03	16.95	22.92	**25.01**	50.00
Labor	2953	18	67.87	43.71	**74.13**	70.49	30.56	32.88	56.08
Labor2	5443	17	2.86	-3.17	0.00	1.59	13.11	**14.28**	10.82
Laheart	200	16	22.08	-0.87	-0.55	3.61	52.44	**52.76**	60.34
Medicare	4406	21	12.48	7.75	21.71	12.4	21.77	**28.79**	17.98
MPG2001	852	10	15.02	-4.1	**37.2**	29.01	34.82	36.25	51.52
Mussels	201	4	44.83	4.39	28.68	25.39	59.15	**61.3**	60.62
Ozone	330	8	13.02	1.77	9.49	9.49	**42.68**	41.01	43.95
Pole	5000	26	25.81	11.37	34.02	**64.47**	34.33	43.11	16.83
Price	159	15	43.72	4.16	29.54	17.73	70.65	**71.08**	64.26
Rate	144	9	-5	-37.5	-12.5	-37.5	18.18	**19.44**	19.71
Rice	171	15	33.72	-1.67	14.84	17.16	59.87	**60.05**	30.91
Rosetta	213	13	50.72	51.06	51.06	55.84	83.25	**87.14**	78.73
Servo	167	4	20.81	-33.33	**56.57**	51.52	28.18	31.1	14.58
Smsa	141	10	26.03	6.03	-33.29	-51.97	84.34	**85.9**	31.92
Soil WC	210	10	3.26	2.17	8.7	18.48	47.87	**48.11**	24.76
Spouse	11136	21	12.9	11.83	36.56	15.59	45.27	**52.11**	34.26
Strike	625	5	-24.13	-1.18	-0.77	-0.35	30.18	**47.93**	13.07
TA	324	6	8.66	0.00	-1.22	-1.22	**36.46**	33.04	31.25
TBI	2159	16	35.51	13.71	33.14	14.95	67.18	**69.41**	68.22
Tecator	215	10	40.62	0.16	19.35	-14.15	63.02	**65.1**	67.64
Tree	100	8	17.68	7.92	-7.23	-10.82	59.22	**61.73**	38.99
Triazine	186	28	25.24	1.51	13.44	12.89	23.49	**25.98**	33.19
Wage	3380	13	12.2	9.15	25.42	11.86	21.31	**38.45**	28.21
Yacht	308	7	-2.19	-5.93	-2.68	**69.65**	43.81	45.1	51.97
Average	–	–	18.41	4.94	20.18	14.6	39.89	**42.89**	39.39

Boosting Method (GBM)[Friedman 2002]. We did not try other methods such as random forest and neural network since Chipman in [Chipman et al. 2012] clearly stated those methods have significantly lower performances compared to BART, which CPXR already outperformed.

To have more confidence in the superiority of our results, we tried to select diverse datasets in terms of the number of predictor variables and the number of instances. The number of predictor variables is ranging from 4 to 28, and the number of instances varies from 96 to 21252. We removed datasets with less than 3 predictor variables.

We implemented the main parts of the CPXR algorithm in Java/maven. We also used some of the R packages in the code mostly to handle traditional modeling jobs. We used Rserve [Urbanek 2003] to communicate between R and Java. For LR, we used R implementation and no parameters. For PLR, we used an R package called "Segmented" [Muggeo 2008] and used the default parameter setting. The number of breakpoints is determined by an auxiliary function called seg.control. For SVR, we used a function called svr with the RBF kernel and other default parameter settings in the "e1071" package [Meyer et al. 2012]. For BART, we used an R package called "BayesTree" [Chipman and McCulloch 2009] and parameters list is determined using cross validation. For regularized regression, we used an R packages called "glmnet" [Friedman et al. 2009] and set $\lambda = 0.05$ in our experiments. For CPXR, we used the fixed parameter setting, $minSup = 0.02$ and $\rho = 0.45$. We used different settings in experiments related to the impact of parameters on the performance.

4.4.2 Prediction Accuracy Evaluation

We used relative RMSE reduction to show how much CPXR and other state-of-the-art regression methods can improve the RMSE over the linear regression. Relative RMSE reduction is defined as

$$\frac{RMSE(LR) - RMSE(X)}{RMSE(LR)} \tag{4.8}$$

where $RMSE(LR)$ is the RMSE of linear regression and $RMSE(X)$ is the RMSE of the competing methods including PLR, SVR, BART, GBM, CPXR(LL), CPXR(LP) and CPXR(LL:Regularized). CPXR(LL) and CPXR(LP) mean the baseline methods are linear regression and piecewise regression respectively. Table 4.3 shows the relative RMSE reduction on the 50 real datasets. Each bold number indicates which method returns the best performance. we excluded CPXR(LL:Regularized) results from the comparison because other methods are not regularized, and the comparison is not reasonable.

In terms of relative RMSE reduction, CPXR(LP) achieved the highest average (42.89%) among other competing methods, which is approximately 24.5%, 38%, 22.7% and 28.29% higher than that of PLR, SVR, BART, and GBM respectively. Counting the number of bold numbers in each column

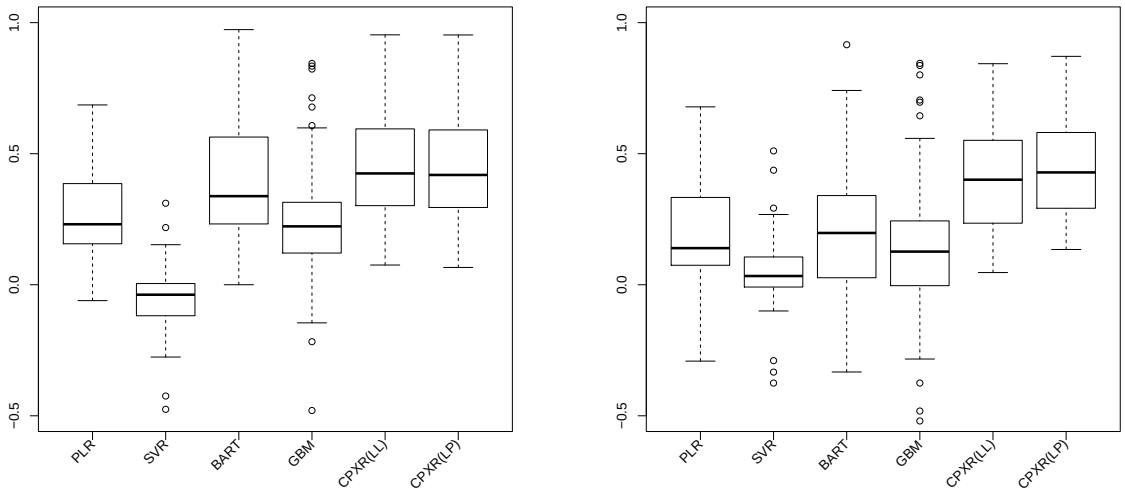
returns CPXR(LP) is the most accurate method in 36 out of 50 datasets, while PLR, SVR, BART, GBM and CPXR(LL) are the best in 0,0,5,4,5 datasets respectively. CPXR(LP) reduced the RMSE of LR for more than 80% on 3 datasets (Budget, Rosetta, Smsa) and more than 70% on 10 datasets. BART improved the RMSE of LR for more than 60% on just two datasets (Budget and Labor). CPXR(LP) and CPXR(LL) are the only methods without negative RMSE improvement among other competing methods. BART could not reduce the RMSE of LR on 9 datasets. GBM and SVR returned negative improvement over LR on 13 datasets. The performance of CPXR(LL:Regularized) is similar to CPXR(LL), although CPXR(LL:Regularized) is a bit less accurate.

It is worth noting that, when CPXR(LP)'s RMSE reduction is the highest, it is usually much larger than that of other methods, and when it is not the highest, it is usually not much smaller than that of other methods (Labor and Servo are two exceptions). Finally, out of 50 datasets, there are 42 datasets where CPXR(LP)'s performance is over 25% (relatively) compared to LR.

Table 4.4: Parameter settings for synthetic datasets

Dataset	D1 D2 D3 D4	D5 D6 D7	D8 D9 D10 D11 D12	D13 D14 D15 D16	D17 D18 D19 D20	D21 D22 D23
Number of instances	5K	5K	5K	1K,5K,10K,20K	5K	5K
Number of variables	10	10	10	10	5,10,15,20	10
Number of patterns	6	6	2,4,6,8,10	6	6	6
Pattern overlap	Small	No,Small,Big	Small	Small	Small	Small
Magnitude of coefficients	1,10,50,100	50	50	50	50	50
Difference in coefficients	3	3	3	3	3	1,3,5
Level of noise	0.05	0.05	0.1	0.1	0.05	0.2

Figures 4.3a and 4.3b represent two boxplots for the relative RMSE reduction of CPXRs and 4 other methods on training and testing datasets respectively. It is obvious that the quartiles of CPXR(LL) and CPXR(LP) are both higher than those of the other methods and interestingly the



(a) (b)

Figure 4.3: Box plots of RMSE reductions on 50 datasets (a:training, b:test)

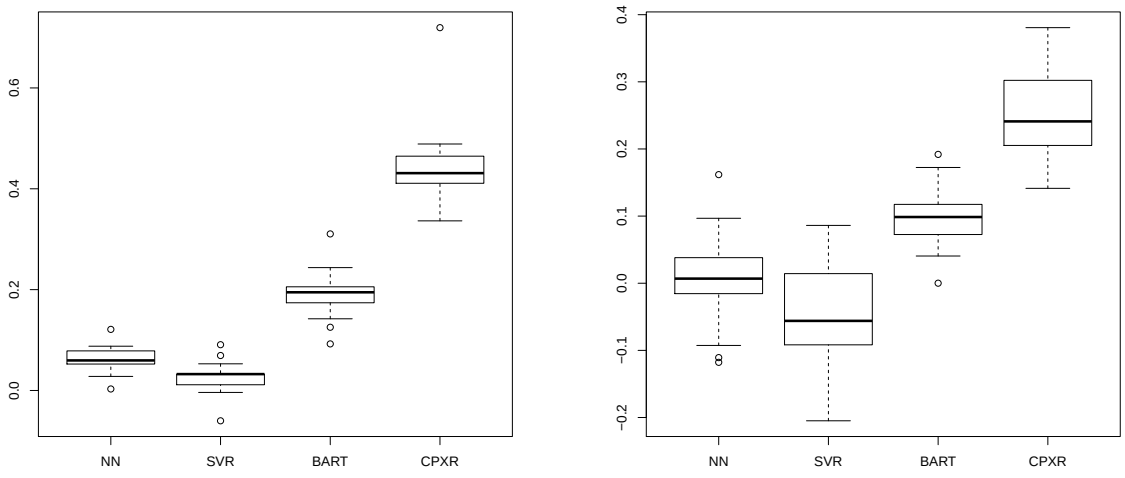
first quartile of CPXR(LL) and CPXR(LP) are higher than the median of the other methods on both training and testing datasets.

Table 4.6 shows (in the "Test" column) the average RMSE reduction over LR on the 23 synthetic datasets (Table 4.4) for PLR, SVR, BART, CPXR(LL) and CPXR(LP) (We used 10-fold cross-validation here). Clearly CPXR outperforms the other methods. On average CPXR(LP) achieved RMSE reduction of 29%, which is about 20% higher than the best competitor methods. These results confirms that the other methods are not capable of modeling diverse predictor-response relationships. Again, PLR's performance is almost identical to that of BART. In all synthetic dataset experiments, we use CPXR default parameters setting ($minSup = 0.02$ and $\alpha = 0.45$).

Figures 4.4a and 4.4b represent the boxplots for the relative RMSE reduction of synthetic datasets and similar to 50 datasets; CPXR has the best performance.

4.4.3 Overfitting and Noise Sensitivity of CPXR

Overfitting is one of the major issues in many of the popular regression and classification techniques. If a regression model overfits, it means the model works well on the training dataset but it performs poorly on test dataset, which means it is not useful in practice. In general, a model is overfitting "if it is more than complex than another model that fits equally well" [Hawkins 2004] and complex models usually perform well but overfit. Simple models such as linear regression perform poor but do



(a) (b)

Figure 4.4: Box plots of RMSE reductions on synthetic datasets (a:training, b:test)

not overfit and are more generalizable. Our goal in this section is comparing the level of overfitting between CPXR and other competing methods.

There are three main reasons that CPXR does not overfit. First of all, CPXR uses simple regression models (LR or PLR) to build local models and those models have less overfitting comparing to other complex models. Second, the number of patterns in CPXR is very small (with average of 7 patterns for real datasets) and then CPXR's models are very simple and easy to understand. Third, it has been discussed extensively in the literature that if the number of predictor variables is more than the number of instances, the risk of overfitting increases. Recall that CPXR removes all patterns whose matching dataset's cardinality is less than the number of predictor variables.

The most common way to quantify model's overfitting, is measuring the "relative accuracy drop" which is how much the accuracy drops from training to testing datasets. The formula to measure relative accuracy drop is the following:

$$\frac{RMSE(training) - RMSE(test)}{RMSE(training)} \tag{4.9}$$

The last column of tables 4.5 and 4.6 demonstrates the magnitude of accuracy drop. Clearly, CPXR(LL) and CPXR(LP) have the smallest drop for 50 datasets and got the second rank after SVR for the synthetic datasets.

Sensitivity to noise is another important issue that affects the performance of regression and

Table 4.5: Performance comparison, and average relative accuracy drop, on 50 datasets

	Average RMSE reduction over LR		**Drop in**
Method	**Training**	**Test**	**accuracy**
PLR	37.11%	18.76%	49%
SVR	7.65%	4.8%	37%
BART	41.02%	20.15%	51%
CPXR(LL)	51.4%	39.88%	22%
CPXR(LP)	53.85%	42.89%	21%

Table 4.6: Performance comparison, and average relative accuracy drop, on synthetic datasets

	Average RMSE reduction over LR		**Drop in**
Method	**Training**	**Test**	**accuracy**
PLR	17%	9.4%	44%
SVR	3.5%	2.5%	29%
BART	19%	9.5%	50%
CPXR(LL)	49.2%	28.4%	42%
CPXR(LP)	49.9%	29.83%	40%

classification models and leads to overfitting. Some regression models are too sensitive to the noisy training dataset and perform poorly on the clean test data. In this scenario, the algorithm tries to fit a model to all instances, including the noisy data points which makes the algorithm very complex.

This section evaluates noise sensitivity of regression algorithms, by examining the difference of modeling' accuracy on noise-added training data and clean test data. This is done by applying classification algorithms including CPXR on 3 real datasets, with noise-added training data and clean test data (With noise level ℓ, we transform the test data by adding a value $z \times y$ to each y value in the original test data, where z is a random value in $[-\ell, \ell]$). We used 0.05, 0.10, 0.15 and 0.20 as noise levels ℓ.

Figure 4.5 shows the drop in accuracy comparing to clean test data for BART, CPXR and GBM models. Apparently GBM is the most sensitive algorithm to noise and CPXR is the least. For example, when we add 5% noise, the RMSE increased by 8%, 12% and 16% for CPXR, BART, and GBM models respectively.

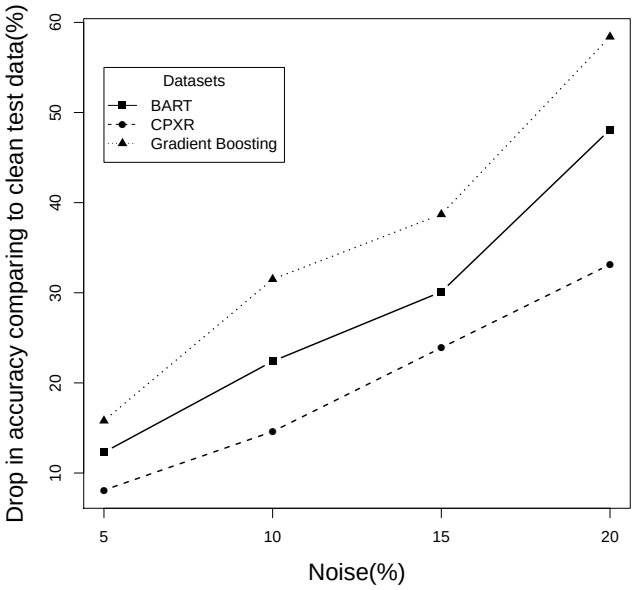


Figure 4.5: Noise sensitivity of regression methods

4.4.4 Data Characteristics and CPXR's Performance

As we discussed earlier in section 4.4.2, CPXR outperforms on most of the datasets but the magnitude of improvements differ. In this section, we want to analyze datasets to understand what are the data characteristics that highly correlated with CPXR's performance. To perform this task, we selected 9 real datasets and divided into two groups: the "high" group consists of five datasets where CPXR has very large RMSE reduction over LR (between 57% and 84% for CPXR), and the "low" group consists of 4 datasets where CPXR has relatively small RMSE reduction (between 16% and 45% for CPXR). Table 4.7 represents the characteristics of these datasets, including number of contrast patterns (equivalence classes) after applying *minSup*, the number of positive improvement patterns (PIPs) (after removing, from the set of patterns above, any pattern P whose R^2 improvement over the baseline model is less than 5%), coverage on large error instances (by PIPs), coverage on all instances (by PIPs), average R^2 improvement of local models of the PIPs, and the difference (given as a ratio) between largest coefficients of the local regression models in the results computed by CPXR. The R^2 improvement by a pattern P is defined as $\frac{R^2(f|_{\mathsf{mds}(P)}) - R^2(f_P)}{R^2(f|_{\mathsf{mds}(P)})}$, where $f|_{\mathsf{mds}(P)}$ denotes f restricted to $\mathsf{mds}(P)$.

What we understand from Table 4.7 is that the number of PIPs, the coverage on large error instances and the difference in the largest coefficients are the most important factors in the CPXR's

performance. These findings confirm the initial idea of CPXR methodology. As we discussed in the introduction, CPXR identifies instances with large error and then tries to characterize those instances using patterns and finally reduces residual error by returning a set of specialized local models. If the number of PIPs is small or the coverage on large error instances is small, then the dataset does not have significant diverse predictor-response relationships; CPXR will not give large improvements in such cases.

Table 4.7: Characteristics of datasets where CPXR has different performance

	Dataset	**# of patterns**	**# of PIP**	**Cov on LE**	**Cov on all data**	**Avg R^2 improvement**	**Difference in coefficients**
High	CPS95	2443	1720	91%	89%	14%	2.1
	Smsa	40	39	87%	85%	24%	2.6
	Price	351	227	95%	79%	11%	2.7
	CPU	138	93	95%	92%	17%	3.2
	Tree	33	16	50%	63%	16%	2.1
Low	Fat	1135	1086	29%	30%	14%	1.1
	Wage	2969	208	34%	57%	4.5%	1.4
	Attend	1402	63	29%	42%	14%	1.7
	Strike	54	48	38%	17%	59%	1.9

In general, we see that all of the datasets in the "high" group have Cov LE$\geq$ 50%, Cov all$\geq$ 63%, and Difference in coefficients$\geq$ 2.1, which all of the datasets in the "low" group have Cov LE$\leq$ 38%, Cov all$\leq$ 57%, and Difference in coefficients$\leq$ 1.9. When datasets have high Cov LE, most of the large error instances are covered by PIPs; when Difference in coefficients is large, the baseline regression models tends to make very large prediction errors on many instances; both offer opportunity for CPXR to make large improvements in prediction accuracy.

4.4.5 Analysis of CPXR's Parameters

In this section, we discuss about the impact of the following parameters on the CPXR's performance:

- Impact of $minSup$
- Impact of ρ
- Impact of the baseline modeling method

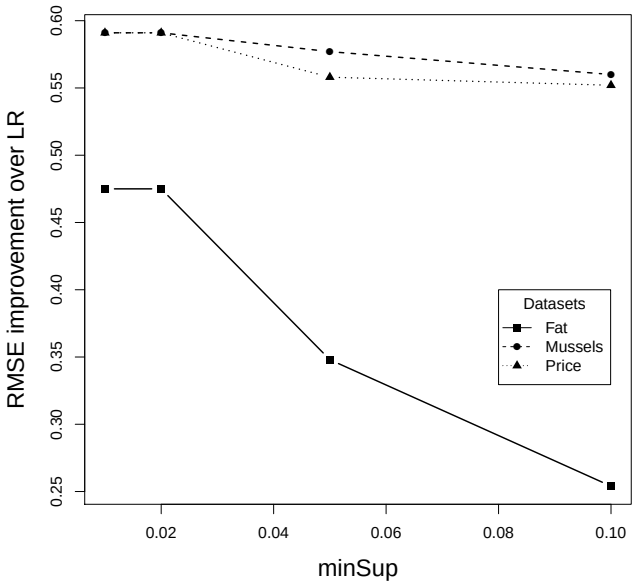


Figure 4.6: minSup's impact on RMSE reduction

- Impact of the number of patterns in PXR models
- Impact of pattern filtering

4.4.5.1 Impact of *minSup*

minSup is one of the two CPXR parameters and is equal to the threshold on the minimum support (in LE) of contrast patterns. We designed a simple experiment to understand the sensitivity of CPXR to different *minSup*s. Figure 4.6 represents the RMSE reduction when the *minSup* varies from 0.01 to 0.1. It is obvious that increasing the *minSup* decreases the number of extracted contrast patterns and as we discussed in the previous section, it may decrease the CPXR performance. Therefore, smaller *minSup* is more desirable. Although $minSup = 0.01$ gives slightly more RMSE reduction than $minSup = 0.02$, $minSup = 0.01$ can lead to significant increase in computing time and memory usage and we decided to use 0.02 in the experiments.

4.4.5.2 Impact of ρ

Another CPXR parameter is ρ which determines the splitting point between LE and SE instances. Similar to the impact of *minSup*, we changed the ρ from 0.4 to 0.7 (*minSup* is fixed at 0.02) in order to find an optimal ρ value. Figure 4.7 represents any ρ value between 0.45 and 0.65 are close

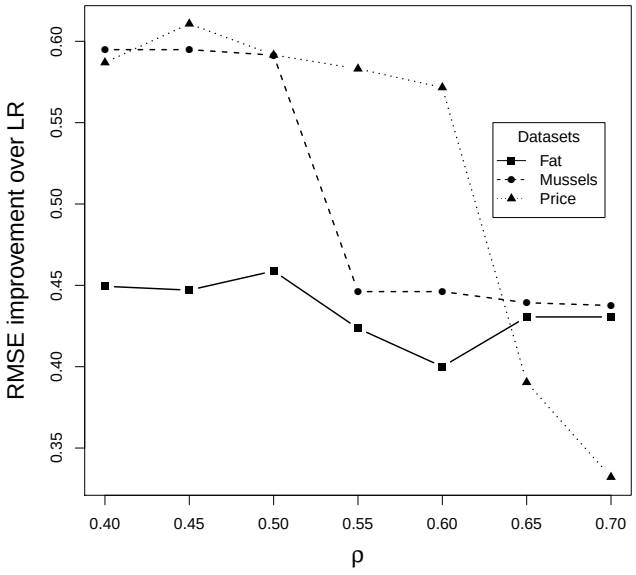


Figure 4.7: ρ's impact on RMSE reduction

to optimal. We used $\rho = 0.45$ for other experiments but recommend to try multiple ρ values.

4.4.5.3 Impact of Baseline Modeling Method

We used linear regression as baseline modeling method in all other experiments. In this part, we want to know how CPXR performs if we change the baseline modeling method. We do not change the local modeling method and is still linear regression. Table 4.8 demonstrates CPXR's performance on 4 real datasets and one synthetic dataset when the baseline modeling methods are LR, BART, GBM, and PLR. It turns out that the results of using BART to generate the baseline models are only slightly better than that of using LR, the results of using PLR to generate the baseline models are almost the same as that of using LR, and the results of using GBM, are actually a bit worse than that of using LR. We conjecture that a possible reason is that the data where GBM makes large errors are overly fragmented, and, as a result, the contrast patterns of LE are not as useful as those for the LR and BART cases. While using BART may also fragment the large error data, the better performance of the BART models (as baseline models) may have compensated the performance of the computed PXR model.

Table 4.8: RMSE of CPXR using different baseline modeling methods

Dataset/Baseline	LR	BART	GBM	PLR
Enroll	0.09	0.06	0.1	0.09
Diabetes	0.72	0.63	1.15	0.46
CPU	10.24	10.09	15.34	11.1
Insur	2.16	2.17	2.27	2.28
D21	3173	3107	3301	3124

4.4.5.4 Impact of Number of Patterns

The number of patterns is determined by the CPXR algorithm. CPXR stops to pick more patterns when adding more patterns does not improve the accuracy significantly. In this experiment, we want to know what is the optimal number of patterns in PXR models. Figure 4.8 shows the impact of the number of patterns on the CPXR performance. In the x-axis, the number of patterns is pre-determined and varies between 5 and 20 and in the y-axis, the CPXR performance is shown. It is clear that the curves are near peak before k is 10. In our experiments on the 50 datasets, the maximum, minimum and average of k are 12,3, and 7 respectively.

4.4.5.5 Impact of Pattern Filtering

Experiments on several datasets indicates that the pattern filtering step in CPXR algorithm shortened the computation time fairly significantly, and no significant loss of prediction accuracy was observed. Pattern filtering often removes more than 20% of the equivalence classes of contrast patterns. For example in Spouse dataset the filtering removed 128240 patterns from a total of 520000. This filtering reduces the time for building the local models and for searching for PXR models significantly and as a result reduces the computation time by at least 20%.

4.4.6 Running Time and Memory Usage

We tested CPXR algorithm on a single processor machine (with a 2.26GHz CPU, 4GB of RAM), concerning running time and memory usage. Table 4.9 shows the results of running time and memory usage on datasets with different sizes. Three important factors that affect the running time are: the numbers of instances, the number of variables, and the number of positive improvement patterns (PIPs, see §4.4.4). Clearly, CPXR built the PXR models within a reasonable amount of time and memory. For 4 of the 6 datasets, the computing time was at most 0.3 minutes. For Pole the computing time is about 8 minutes; this could be attributed to the large numbers of PIPs and

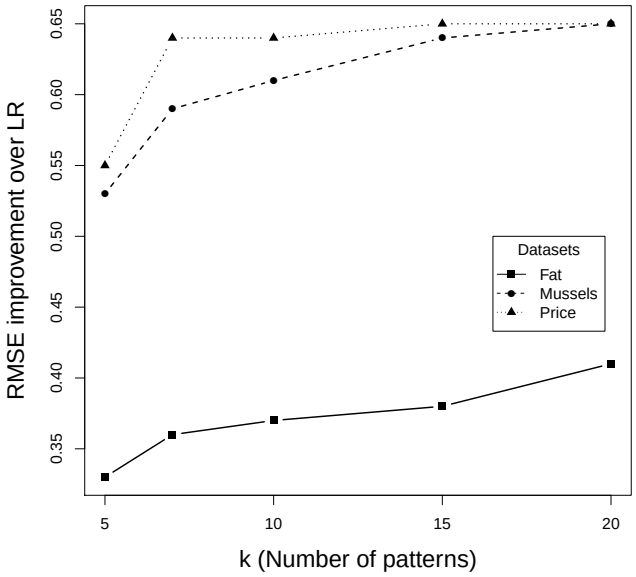


Figure 4.8: Impact of the number of patterns on CPXR's performance

variables.

Table 4.9: Running time and memory usage of CPXR, and running time of other algorithms

Dataset	**# of instances**	**# of attributes**	**# of PIPs**	**CPXR time**[a]	**CPXR memory**	**LR time**[b]	**PLR time**[b]	**SVR time**[b]	**BART time**[b]	**GBM time**[b]
Fat	252	14	1135	0.1	1.1	0.25	0.22	0.058	9.78	1.06
Mussels	201	4	38	0.015	0.85	0.29	0.15	0.036	7.56	0.68
Price	159	15	362	0.022	0.8	0.27	0.25	0.042	6.89	1.09
Spouse	11136	21	228165	6.1	48	0.8	0.75	39.26	1839	9.19
Pole	5000	26	4442542	8.1	83	0.54	0.79	16.12	14.5	5.57

a: minues; b: seconds.

4.4.7 Conclusion

In this chapter, we introduced the diverse predictor-response relationship phenomenon, which says in numerical prediction problems the predictor-response relationships fitting different logical groups of data are often highly different. We also introduced a novel type of regression models, called pattern aided regression (PXR) models, defined using small sets of patterns and corresponding local regression models. PXR models can naturally model diverse predictor-response relationships. The

paper introduced a regression method (CPXR) for building highly accurate and interpretable PXR models, which outperforms state-of-the-art regression methods, often by big margins, in experiments. PXR models are easy to interpret, and they achieve highly accurate prediction modeling with low model complexity; the above indicates that the patterns and local regression models construct of PXR models are very powerful, enabling PXR models "to achieve more with less". PXR and CPXR is especially effective for high-dimensional applications involving multiple multivariable interactions.

5

Contrast Pattern Aided Classification (CPXC)

In this chapter, we introduce the Pattern Aided Classification (PXC), a new type of classifier, which uses several pattern and local classifier pairs. We also introduce a new classification algorithm, called Contrast Pattern Aided Classification (CPXC), to build accurate PXCs [Dong and Taslimitehrani 2016]. Experiments show that CPXC outperforms existing classification algorithms consistently, often by wide margins.

5.1 PXC Concepts

The main idea of Pattern Aided Classifier (PXC) is using a set of contrast patterns to represent several subgroups of data and then using a set of local classifiers to classify data instances matching those patterns. The following definitions are required to define PXC concepts.

Definition: Given a training dataset $D = \{(x_i, y_i) \mid 1 \leq i \leq n\}$, a classifier built on D as training data is called *baseline classifier* and is denoted by h_b. h_b can be built by any classification algorithm.

Definition: Given the matching dataset of pattern P in dataset D, $mds(P, D)$, a classifier built on $mds(P, D)$ as training data is called *local classifier* and denoted by h_p. h_p is a classifier associated to pattern p and can be built by any classification algorithm such as logistic regression, decision tree and naive Bayes classifier. We write h_{p_i} as h_i for pattern p_i.

Definition: Given an instance x and classifier h, $h(x, C)$ denotes the *classification score* of h for an instance x on class C. $h(x, C)$ can be in different forms such as probability of classifying x as a member of C.

Definition: Given an instance x and classifier h, $match(x, p)$ denote the *confidence* of "instance x

matches pattern p." In the simplest 0-1 case, $match(x, p)$ is either 0 (not matching) or 1 (matching).

We are now ready to define the central concept of this chapter, namely pattern aided classifiers. We limit ourselves to binary classification in this chapter.

Definition: A *Pattern Aided Classifier (PXC)* is given by a tuple $M = ((p_1, h_{p_1}, w_1), ..., (p_k, h_{p_k}, w_k), h_d))$, where $k > 0$ is an integer, $p_1, ..., p_k$ are patterns, $h_{p_1}, ...h_{p_k}$ are local classifiers, h_d is the default classifier, and $w_1, ..., w_k$ are weights. The classification score of CPXC for an instance x on class C_j is defined as

$$weighted\_votes(M, C_j, x) = \begin{cases} \frac{\Sigma_{P_i \in \pi_x} w_i \times h_{p_i}(x, C_j)}{\Sigma_{P_i \in \pi_x} w_i} & \text{if } \pi_x \neq \emptyset \\ h_d(x) & \text{otherwise} \end{cases} \quad (5.1)$$

where $\pi_x = \{P_i \mid 1 \leq i \leq k, x \text{ satisfies } P_i\}$.

Since the patterns in a PXC are used as conditions, PXC classifiers can also be called *conditional aided classifiers (CXC)*. It is notable that several previously studied voting based classifiers can be presented as degenerate PXCs or CXCs.

As will be seen later, we use the weight w_i of a local classifier h_{p_i} as confidence score for h_{p_i}'s classification. The above weighted vote formula combines that confidence score with the classification score ($h_{pi}(x, C_j)$) of classifier h_{p_i}.

Table 5.1: A PXC $M_0 = ((p_1, h_1, 0.7), (p_2, h_2, 0.4), h_d)$

p_1	$X \leq 5 \& Y = b_1$
p_2	$X > 4$
h_1	if $A = a_1$ then C_1 else C_2
h_2	if $A = a_1$ then C_2 else C_1
h_d	if $Y = b_2$ then C_2 else C_1

Example: Table 5.1 gives a PXC M_0, where p_i's local classifier is h_i, and h_d is the default classifier. (h_1; h_2; h_d are decision trees shown as rules.) The underlying data have three features, X, Y, and A. How M_0 classifies an instance depends on the number of patterns that the instance matches.

- Matching 0 pattern: For example, M_0 assigns C_2 to $t_0 = (X = 2, Y = b_2, A = a_1)$ following h_d, since t_0 matches neither p_1 nor p_2.
- Matching 1 pattern: For example, M_0 assigns C_1 to $t_1 = (X = 2, Y = b_1, A = a_1)$ following h_1, since t_1 matches p_1 only.
- Matching 2 or more patterns: For example, M_0 assigns C_1 to $t_2 = (X = 5, Y = b_1, A = a_1)$

using weighted voting, since t_2 matches both p_1 and p_2, h_1 assigns C_1 to t_2, h_2 assigns C_2 to t_2, and h_1's weight (0.7) is larger than h_2's weight (0.4).

5.2 The CPXC Algorithm

After presenting the main steps of CPXC, this section gives details of CPXC's techniques. It contains three subsections, one outlining CPXC, one for techniques which are quite similar to those of CPXR, and one for techniques which are fairly unique to CPXC. We note that for CPXC there are more issues as well as more opportunities compared to CPXR, mainly due to the difference between classification and regression.

5.2.1 Main steps of CPXC

Table 5.2 gives an outline of CPXC. The main objective of CPXC is to compute a small cooperating set of patterns, where each pattern characterizes a subgroup of data such that (a) a baseline classifier (built on the given training data) makes large classification errors on data instances in the subgroup and (b) a highly accurate local classifier exists to correct those errors, and (c) the small cooperating set of patterns collectively defines an accurate PXC.

CPXC takes three inputs: a training dataset D, a ratio $\rho > 0$ for dividing D into large error (LE) and small error (SE) parts, and a $minSup$ threshold on the support of contrast patterns in LE.

Table 5.2: Outline of the CPXC algorithm

Input	Training dataset D, ρ and $minSup$
Step 1	Train a baseline classifier h_0
Step 2	Split D into LE and SE based on h_b's error
Step 3	Discretize numerical predictor variables into bins using equi-width and entropy method
Step 4	Perform contrast pattern mining of LE dataset
Step 5	Apply a set of filters to reduce the number of patterns
Step 6	Train local classifiers for the remaining contrast patterns
Step 7	Remove patterns of low utility
Step 8	Select an optimal set of patterns, $PS = \{P_1, ..., P_k\}$
Step 9	Determine weights and local classifiers h_p associated to each pattern in PS
Step 10	Train the default classifier h_d for $D - \bigcup_{i=1}^{k} mds(p_i)$
Output	The PXC $((P_1, h_1, w_1), ..., (P_k, h_k, w_k), h_d)$

5.2.2 Techniques Similar to CPXR

In the first step of CPXC algorithm, CPXC builds a baseline classifier h_0 on D as training dataset. CPXC can use any classification algorithm for building h_0 such as Logistic Regression (Log), Naive Bayesian Classifier (NBC) and Decision Tree (DT) [Quinlan 1993]. CPXC focuses on contrast patterns that occur frequently in the large error (LE) data, and infrequently in the small error (SE) data. Defining LE and SE is needed to allow us to get hold of useful contrast patterns (which are associated with significant opportunities where highly accurate local classifiers can correct h_b's classification errors).

To split D into LE and SE in step 2, we need to find a cut-point κ using the ratio of accumulative classification error of h_b in LE vs accumulative classification error of h_b in D. The cut point κ is determined by

$$\rho \approx \frac{\sum_{x \in D, err(h_b,x) > \kappa} err(h_b, x)}{\sum_{x \in D} err(h_b, x)} \tag{5.2}$$

on the error based sorted data instances (ρ is given by the user.). Then $LE = \{x \in D \mid err(h_b, x) > \kappa\}$ and $SE = D - LE$.

In step 3, CPXC discretizes numerical predictor variables. If entropy based binning method [Fayyad and Irani] cannot find the binning splits, equi-width binning will be used. CPXC does not change the categorical variables. In the next step (step 4), GcGrwoth [Li et al. 2005] is used to mine contrast patterns in LE with the $minSup$.

We use $minSup$ to control our initial pool of contrast patterns. We do not want to set $minSup$ too large, to avoid missing valuable patterns. So the initial pool can be very large. Searching over a large pool for a desirable pattern set is time-consuming, and building local classifiers for all those patterns is prohibitive. It is important to reduce the pool in order to find efficiently a high-quality pattern set to build an accurate PXC. In steps 5 and 7, CPXC uses several techniques toward that goal.

GcGrowth returns the equivalence class (EC) of each pattern, and each EC consists a set of minimal generators (MG). CPXC picks the shortest MGs and ignores other patterns. Since the patterns of each ECs have the same matching datasets and of course the same behavior, this step helps on gaining efficiency while minimizing the loss of useful patterns.

Also, CPXC applies the following filtering techniques to reduce the number of patterns.

- If the number of data instances matching on pattern p (size of matching dataset) is less than the number of predictor variables in D, pattern p is removed from contrast patterns set (CPS). This technique helps to control overfitting as well.

- If the average error reduction (will be defined later) of a pattern is too small, CPXC removes

the pattern (Step 7).

- There are some patterns with the similar matching datasets. As an optional filter, CPXC calculates the Jaccard similarity of each pair of patterns: Given two patterns p_1 and p_2, $JS(p_1, p_2) = \frac{|mds(p_1) \cap mds(p_2)|}{|mds(p_1) \cup mds(p_2)|}$; if $JS(p_1, p_2) > 0.9$, the pattern with smaller average error reduction is removed.

In step 6, CPXC builds local classifiers for all remaining patterns on their matching datasets. In step 8, there is a search process to select a small cooperative set of patterns and build a PXC. The objective function used in this step will be discussed later. The iterative search process involves two nested loops. Each iteration of the outer-loop selects a pattern p that maximizes the *obj* function to add to the current pattern set PS. After each addition, CPXC uses an inner-loop to repeatedly selects a pattern pair q, p_q where $q \in PS$ and $p_q \in CPS - PS$, such that replacing q by p_q gives the largest increase to the $obj(PS)$; the replacement is made if the replacement actually increases the *obj* value. The inner-loop terminates if its last iteration did not add more than 0.1% of the *obj* function value, and the outer-loop terminates if its last iteration did not improve the *obj* function value by more than 1%. The number of patterns in the PXC is determined by the algorithm - it is the number of patterns in PS when the search process converges. If simplicity is a criterion, the number of patterns can be given as a parameter. In experiments, the PXCs produced by CPXC contain 9.6 patterns on average.

Local classifiers and weights associated to each selected pattern is determined in step 9. Some of the data instances are not covered by any of the patterns in PS. CPXC trains h_d on those not covered instances in step 10 and returns PXC as a set of patterns, local classifiers, weights and default classifier.

Most of the techniques discussed so far are similar to CPXR. In the following parts, we discuss details of the main steps of CPXC whose techniques are substantially new.

5.2.3 Techniques Unique to CPXC

5.2.3.1 Measuring Classification Error

A key idea of CPXC is to identify patterns p such that the baseline classifier h_b makes large classification errors on instances in $mds(p, D)$ and many of those errors can be corrected by a local classifier. One important issue is the loss function that is used to measure classification errors. Let h denote a classifier, x a data instance, and y the class label of x. We considered three approaches.

- The *binary error* measure is defined by

$$error_b(h, x) = \begin{cases} 1 & \text{if } h(x) = y \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

- The *probabilistic error* measure is defined by

$$error_p(h, x) = |y - h(x)| \tag{5.4}$$

  where it is assumed that $y \in \{0, 1\}$ and $h(x)$ is h's predicted probability for "x's class label is 1".

- The *standardized error* (Pearson residual) is defined by

$$err_s(h, x) = \frac{y - h(x)}{\sqrt{h(x)(1 - h(x))}} \tag{5.5}$$

  where y and $h(x)$ are same as for equation 5.4. This is a popular measure for goodness of fit of logistic regression models.

The winner is the probabilistic error measure. Because it allows us to differentiate the degree of classification errors and hence finding contrast patterns that pay more attention to more important errors.

5.2.3.2 Choosing Methods for Baseline Classifiers and Local Classifiers

The choice depends on the different requirements for building those classifiers, regarding their purposes and computational needs. First, the baseline classifier h_b serves as a starting point for CPXC, helping towards mining contrast patterns that characterize the data instances (in LE) where h_b makes (large) classification errors. It is discarded once CPXC finds a PXC. A desirable h_b should (a) be accurate and (b) (more importantly) allow the discovery of diverse contrast patterns p (c) that occur more often in LE than in SE (d) whose local classifier h_p can correct many of the classification errors of h_b on $mds(p, D)$. Second, the local classifiers should be interpretable since they are in the final PXC. Third, since CPXC needs to learn local classifiers for many contrast patterns, the algorithm for learning local classifiers must be very fast. In contrast, the baseline classifier is computed just once, so the algorithm does not need to be very fast.

In this study, we evaluated six classification algorithms for learning baseline, local and default classifiers, including Logistic Regression (Log), Naive Bayesian Classifier (NBC), Decision Tree (DT), AdaBoost, Support Vector Machine (SVM), and Random Forest (RF). Since the first three are fast, and the classifiers they built are interpretable, we used them for learning both baseline and

local/default classifiers. The last three are slow, and their classifiers are not highly interpretable. Therefore, we only used them for learning baseline classifiers.

Experiments show that there is some dependency between the methods for building the baseline and local classifiers (see Section 5.3).

5.2.3.3 Quality Measures on Pattern Sets

We now present three measures to determine the quality of pattern set, which will serve as objective functions for the pattern set search. Let $PS = \{p_1, ..., p_k\}$ be a pattern set, h_b be a baseline classifier, and $w_i = w(p)$ be a function for determining the weight of p in a PXC model. Let $M = ((p_1, h_1, w_1), ..., (p_k, h_k, w_k), h_b)$ denote a PXC model constructed from PS, h_b. Let err be a measure for classification errors.

The *total error reduction (TER)* of PS is defined by

$$TER(PS) = \frac{\Sigma_{x \in \mathsf{mds}(PS)} |err(h_b, x) - err(M_{PS,h_b}, x)|}{\Sigma_{x \in D} err(h_b, x)} \tag{5.6}$$

where $\mathsf{mds}(PS) = \cup_{i=1}^{k} \mathsf{mds}(p_i)$. The other two measures are the ACC (*accuracy*) of M_{PS,h_b} and the AUC of M_{PS,h_b}.

Experiments show that TER is the best objective function. We believe this is because TER measures the amount of corrected classification errors (of the baseline classifier) while the others do not: TER helps picking patterns that correct classification errors, while ACC or AUC may mislead the algorithm to pick patterns whose matching datasets are already easy to classify by the baseline classifier.

5.2.3.4 Quality Measures on Patterns, Weights and Local Classifiers

To measure the utility of a pattern p, we can use the *accuracy* or AUC of h_p. A third option is using the *average error reduction (AER)* of p, defined as

$$AER(p) = \frac{\Sigma_{x \in \mathsf{mds}(p)} |err(h_b, x) - err(h_p, x)|}{\Sigma_{x \in \mathsf{mds}(p)} err(h_b, x)}.$$

These measures are used to remove "useless" patterns after local classifiers are built. It turns out that using *accuracy* or AUC are not as good as using AER, since a pattern p can have high ACC or AUC because data in $\mathsf{mds}(p, D)$ are easy to classify and the baseline classifier is already very accurate on $\mathsf{mds}(p, D)$ (hence p is not useful for producing an accurate PXC classifier). All of the above three measures can be used as weights of patterns.

5.2.3.5 Using the Confidence of Match to Classify Instances

In chapter 3, we discussed matching instance x to the pattern p and used it as binary value to express matchness of an instance to a pattern such as "an instance x matches a pattern p" or "an instance x does not match a pattern p". A more general approach is to use a range of values between 0 and 1 to express the confidence of a match.

Let $match(x, p) > 0$ denotes the confidence of "instance x matches pattern p", M be a PXC, p_i be a pattern occurring in M, and x be a data instance. We note that p_i is actually just a representative of the $EC(p_i) = \{q \mid mds(q) = mds(p_i)\}$ set of patterns. Since h_{p_i} is a classifier built to classify all data instances in $mds(q) = mds(p_i)$ for every $q \in EC(p_i)$, p_i is not more special than the other MGs of $EC(p_i)$ when considering matching x with $EC(p_i)$. In practice, x may match some MG of $EC(p_i)$ but x does not match p_i, especially when x is new (not in the original training dataset D). In general, x may match several MGs of $EC(p_i)$.

We now provide some details on ECs and MGs to help understand the situation. It can be proven that $EC(p_i)$ is precisely the set of patterns q satisfying q is a subset of $closed(p_i)$ and q is a superset of some $p \in MG(p_i)$, where $closed(p_i)$ denotes the closed pattern of $EC(p_i)$ and $MG(p_i)$ denotes the set of minimal generators of $EC(p_i)$. In general, $MG(p_i)$ can contain a fairly large amount of MGs; e.g., for the dataset $D = (abcdeghi, avdg, bcdghi, abdhi, bceghi\}$ of five transactions, the EC of $bcdghi$ contains six MGs (namely, $bcd, bdg, cdh, cdi, dgh, dgi$). Each of those MGs presents a unique minimal combination of single-item conditions that differentiates this EC from other ECs.

The new way to define $match$ is given by

$$match_c(x, p_i) = \frac{|\{q_i \in MG(p_i) \mid x \quad matches \quad q_i|}{|MG(p_i)|} \tag{5.7}$$

Here, all MGs in $MG(p_i)$ are treated as equal and $match_c(x, p_i)$ represents the confidence of "x matches the EC of p_i". The more MGs x matches, the higher the confidence of the match.

For completeness, we give the "binary" definition of $match$: $match_b(x, p_i) = 1$ if x matches p_i and $match_b(x, p_i) = 0$ otherwise. Here, p_i is treated as more important than the other MGs in $MG(p_i)$.

Now, we can define PXCs based on the new $match$'s definition:

Definition: Given a PXC classifier $M = ((p_1, h_{p_1}, w_i, ..., (p_k, h_{p_k}, w_k), h_d)$ and an instance x, the *classification score* of CPXC for an instance x on class C_j is defined as

$$weighted\_votes_m(M, C_j, x) = \begin{cases} \frac{\Sigma_{P_i \in \pi_x} w_i \times match(x,p_i) \times h_{p_i}(x, C_j)}{\Sigma_{P_i \in \pi_x} w_i \times match(x,p_i)} & \text{if } \pi_x \neq \emptyset \\ h_d(x) & \text{otherwise} \end{cases} \tag{5.8}$$

where $\pi_x = \{P_i \mid 1 \leq i \leq k, match(x, p_i) > 0\}$.

5.3 Experimental Evaluation (CPXC)

This section presents the experimental evaluation of CPXC. The experiments

- use 17 benchmark datasets from the UCI Machine Learning Repository [Bache and Lichman 2013],
- compare CPXC with 8 popular classification algorithms (including variants),
- consider several quality measures, including AUC, overfitting prevention, noise resistance,
- investigated the impact of several parameters including $minSup$, ρ, loss functions, objective functions, weights and the number of patterns, and
- measured CPXC's running time and memory usage on some of the benchmark datasets.

5.3.1 Datasets and experiment settings

The datasets used in our experiments are all from UCI [Bache and Lichman 2013]. Table 5.3 represents some characteristics of datasets including dataset name, the number of instances and number of attributes. The first 8 datasets are considered "hard" and the last 9 are considered "easy" datasets. A dataset is considered hard if none of the competing classification methods considered in our experiments cannot return classifiers with AUC more than a specified threshold (e.g. 0.76 as used here). If the dataset is not considered hard, then it is considered easy. The class labels of all datasets are binary. Poker class labels are not binary and then we removed instances whose classes are not in $\{0, 1\}$ and the number of instances dropped to 948717.

We compared the performance of CPXC to a set of popular classification methods and used their R implementations to calculate AUC. We used rpart [Therneau et al. 2010] for C4.5 [Quinlan 1993], e1071 [Meyer et al. 2012] for Naive Bayes (NBC), glm for logistic regression (Log), randomforest [Liaw and Wiener 2002] for Random Forest (RF) [Breiman 1996], e1071 [Meyer et al. 2012] for SVM [Vapnik 2013], and gbm [Ridgeway 2006] for boosting [Freund and Schapire 1997]. For SVM, we used SVM(Lin), SVM(Poly) and SVM(RBF) to respectively denote SVM with linear, polynomial and radial basis function kernels. GBM in R is a generalization of AdaBoost. We used the mdlp function in the discretization package of R (using default termination-condition setting) for entropy based binning, and we set the number of bins to 4 for equi-width binning. We used default parameter settings for all these implementations, as it is not feasible to fine tune parameter settings in cross validation studies involving multiple datasets.

We use CPXC(X-Y) to denote CPXC where X and Y are algorithms used to build the baseline and local classifiers respectively. 7-fold cross validation is used in our experiments. CPXC parameters

were set to $minSup = 0.02$ and $\rho = 0.45$. CPXC used TER as the objective function and AER for pattern filtering and weighting local classifiers. The loss function is used in our experiments is probabilistic. In this section, CPXC refers to CPXC(NBC-DT).

5.3.2 CPXC's Performance vs Other Algorithms

Comparison of the algorithms on Group-A datasets: Table 5.3 represents the AUC of CPXC and 8 other classification algorithms. The comparison reveals several important points. First, CPXC achieved average AUC of 0.886 on the 8 Group-A datasets; the best performing traditional algorithm is RF, which obtained average AUC of 0.638. Second, comparing CPXC with RF on a per dataset basis, CPXC outperformed RF on AUC by 0.248 on average, CPXC's AUC is never lower than RF's AUC on these 8 datasets. Third, the minimum AUC of CPXC on these 8 datasets is 0.85. In contrast, with the exception of just one algorithm-dataset pair (where the AUC is 0.76), the 8 traditional algorithms all failed to build accurate classifiers with AUC higher than 0.71.

The results show that CPXC builds significantly more accurate classifiers on datasets that are challenging to traditional classification algorithms. We believe this happened because these datasets contain highly heterogeneous subgroups, CPXC can effectively handle that kind of heterogeneity and traditional algorithms cannot.

Comparison of the algorithms on Group-B datasets: Table 5.3 indicates two points. First, CPXC achieved average AUC of 0.983 on the 9 Group-B datasets, while the best performing traditional algorithms (Boosting and RF) obtained average AUC of 0.968 and 0.966 respectively. Second, comparing CPXC with Boosting on a per dataset basis, CPXC outperforms Boosting on AUC by 0.016 on average; CPXC's AUC is never lower than that of Boosting by more than 0.01. Based on the above, it is clear that CPXC typically builds more accurate classifiers on the Group-B datasets than traditional algorithms, although it does not have a significant advantage here.

Tables 5.5 and 5.4 show CPXC's ACC is better than the best reported by [Fernández-Delgado et al. 2014] by 5.56% on average of some of the easy datasets.

5.3.3 CPXC's Noise Sensitivity and Overfitting

Overfitting is one of the major issues in many of the popular regression and classification techniques. If a regression model overfits, it means the model is perfectly accurate on training dataset, but it poorly performs on test dataset and then it is not useful in practice. This section compares the classification algorithms on three perspectives concerning overfitting.

- In general, a model is overfitting if it is more complex than other models[Hawkins 2004]. As discussed above, classifiers built by CPXC are often more accurate than those built by other

Table 5.3: Comparison on AUC of CPXC vs 8 algorithms. The 8 Group-A datasets are at the top half, and the 9 Group-B datasets are at the bottome half

Dataset	**# of inst.**	**# of attr.**	**Boosting**	**DT**	**NBC**	**Log**	**RF**	**SVM (Lin)**	**SVM (Poly)**	**SVM (RBF)**	**Max of 8**	**CPXC (NBC-DT)**
Planning	182	12	0.521	0.504	0.612	0.304	0.373	0.634	0.634	0.617	0.634	0.89
Monk 2	432	6	0.61	0.49	0.58	0.52	0.71	0.49	0.63	0.57	0.71	0.9
ILPD	583	10	0.7	0.58	0.63	0.67	0.71	0.64	0.66	0.51	0.71	0.91
Blood	748	4	0.69	0.7	0.65	0.67	0.71	0.68	0.67	0.62	0.71	0.88
Poker	948717	10	0.6	0.6	0.5	0.5	0.76	0.5	0.51	0.63	0.76	0.85
HillValley	606	100	0.5	0.63	0.65	0.66	0.6	0.67	0.67	0.48	0.67	0.89
StatLog	690	14	0.67	0.66	0.62	0.6	0.66	0.64	0.63	0.49	0.66	0.91
Congress	435	16	0.58	0.66	0.6	0.57	0.58	0.58	0.58	0.58	0.6	0.86
Average (hard)			0.621	0.603	0.605	0.562	0.638	0.604	0.623	0.562	0.638	0.886
EEG	14980	14	0.97	0.73	0.7	0.67	0.96	0.68	0.68	0.96	0.96	0.99
Monk1	432	6	0.98	0.91	0.87	0.84	0.98	0.74	0.91	0.93	0.98	0.98
Monk3	432	6	0.99	0.98	0.91	0.92	0.99	0.91	0.91	0.98	0.99	0.99
Climate	540	17	0.96	0.81	0.9	0.94	0.97	0.98	0.98	0.98	0.98	0.97
Banknote	1372	4	1	0.98	0.98	0.99	1	0.99	0.99	0.99	1	1
Bank marketing	45211	16	0.94	0.87	0.91	0.92	0.93	0.91	0.91	0.9	0.93	0.99
WholeSale	440	7	0.97	0.96	0.97	0.96	0.98	0.96	0.96	0.98	0.97	0.96
Mammography	961	5	0.94	0.91	0.94	0.94	0.93	0.93	0.93	0.92	0.94	0.98
Steel	1941	26	0.96	0.88	0.91	0.95	0.95	0.94	0.94	0.91	0.95	0.99
Average (easy)			0968	0.892	0.899	0.903	0.966	0.893	0.912	0.949	0.966	0.983

algorithms. Moreover, classifiers built by CPXC often use a small number of (around 10 on average in the experiments) of pattern and local classifier pairs, the patterns are short and simple, and the local classifiers are simple. So classifiers built by CPXC are accurate and quite simple.

- "AUC drop," the relative difference between the accuracies on training and testing data, is another way to evaluate overfittingness. Table 5.6 shows that CPXC's relative drop in AUC from training to testing data is approximately equal to the lowest among the algorithms.

- Sensitivity to noise can be used for evaluating overfittingness. Here, we examine the relative difference of the accuracy of classifiers built by different algorithms on clean training data and on noise-added test data, on 3 real datasets. (With noise level l, we transform original test data by adding a value $z \times x$ to each value x (of numerical attributes) in the original test data, where z is a random value in $[-l, l]$.) We used 0%, 5%, 10%, 15% and 20% as noise levels l.

Table 5.4: Accuracy of CPXC(NBC-DT) and the best algorithm reported by [Delgado, 2014] on some of the hard datasets

	Planning	**Monk 2**	**ILPD**	**HillValley**	**StatLog**	**Congress**	**Average**
Best of [Fernández-Delgado et al. 2014]	72.8	67.8	77.6	74.3	69.1	63.2	70.8
CPXC(NBC-DT)	85.2	78.1	88.4	83.4	80.8	78.9	82.47
Difference	12.4	10.3	10.8	9.1	11.7	15.7	11.67

Table 5.5: Accuracy of CPXC(NBC-DT) and the best algorithm reported by [Delgado, 2014] on some of the easy datasets

	Bank marketing	**Monk 1**	**Monk 3**	**Average**
Best of [Fernández-Delgado et al. 2014]	90.5	79.9	77.3	82.57
CPXC(NBC-DT)	97.2	86.2	81.3	88.23
Difference	6.7	6.3	4.0	5.67

Table 5.6 shows the relative drop (in percentage) in AUC for classifiers built by the classification algorithms. (15 of the 17 datasets that do not contain categorical attributes were included in deriving the table.)

Based on the above we see that CPXC is the winner on overfitting minimization and noise sensitivity.

Table 5.6: Drop of AUC vs noise levels for various algorithms

Method/Noise	**0%**	**5%**	**10%**	**15%**	**20%**	**Avg**
RF	5.73	6.61	12.48	25.83	33.54	16.84
CPXC	5.87	6.79	12.92	24.7	32.7	16.6
Boosting	7.02	8.93	14.2	26.8	34.65	18.32
Log	7.04	10.56	14.63	24.7	33.94	18.17
NBC	7.06	10.58	15.26	27.89	35.1	19.18
SVM	8.6	10.34	16.28	29.59	38.02	20.57
DT	8.8	11.04	16.78	30.3	43.1	22.00

5.3.4 Impact of Different Baseline/Local Classifications Algorithms on the Accuracy

We conducted experiments to systematically evaluate which combinations of algorithms lead to the best results. Table 5.7 reports the CPXC's AUC for different variants of baseline and local classification algorithms. Table's header represents the abbreviations of baseline and local methods used in the experiment. For example, D-N means the baseline method is decision tree, and the local method is naive Bayes. We used SVM to build baseline classifiers and not used to build local classifiers. In general, with the exception of the Log-Log combination, the average AUC achieved by different combinations of classification algorithms for baseline and local classifier are fairly compatible (within 0.03 of the best AUC). The NBC-DT combination is the consistent winner. NBC builds very simple classifiers and then help to mine more high-quality contrast patterns, DT is quite competitive with NBC and Log with respect to classification accuracy, and NBC and DT are complementary to each other. The Log-Log combination's average AUC is lower than that of the NBC-DT combination by about 0.05 over the 8 Group-A datasets and by about 0.016 on the 9 Group-B datasets.

5.3.5 Impact of Parameters and Techniques on the CPXC's Performance

Impact of $minSup$ and ρ: $minSup$ is the minimum support threshold used in mining contrast patterns in LE. Figure 5.1 represents the CPXC's AUC when the $minSup$ varies from 0.01 to 0.1 (with ρ fixed at 0.45). It is obvious that increasing the $minSup$ decreases the number of extracted patterns and it may decrease the CPXC performance; then less $minSup$ is more desirable. Although $minSup = 0.01$ gives slightly better AUC than $minSup = 0.02$, $minSup = 0.01$ can lead to significant increase in the number of patterns, computing time, and memory usage; therefore we decided to use 0.02 in our experiments.

The other CPXC parameter is ρ (used for dividing D into LE and SE). Figure 5.2 represents how AUC changes when ρ varies between 0.3 and 0.7 ($minSup$ is fixed at 0.02). Clearly, all ρ values between 0.45 and 0.65 are close to optimal. We recommend trying multiple ρ values. We used $\rho = 0.45$ in our experiments.

Impact of the number of patterns: The CPXC algorithm determines the number of patterns in the pattern set. In this experiment, we want to find the optimal number of patterns on some of the real datasets. Figure 5.3 shows the influence of the number of patterns k, when k is predetermined. We see that the curves are near their peak around $k = 10$. In our experiments on 17 datasets, the average of k is 9.6.

Impact of loss function on the CPXC's performance: Figure 5.4 gives us a clear picture that

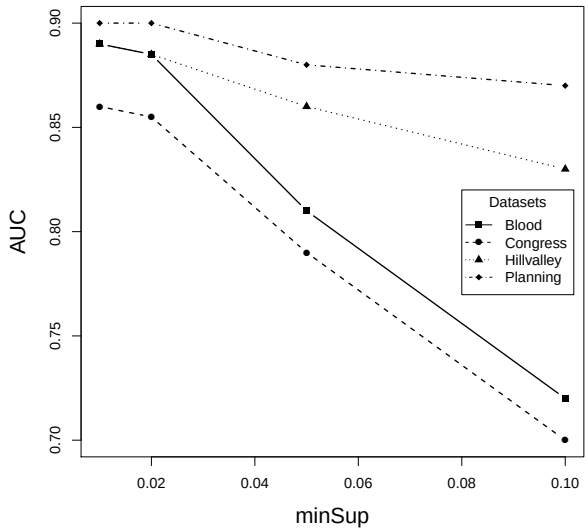


Figure 5.1: Impact of *minSup* on CPXC's performance

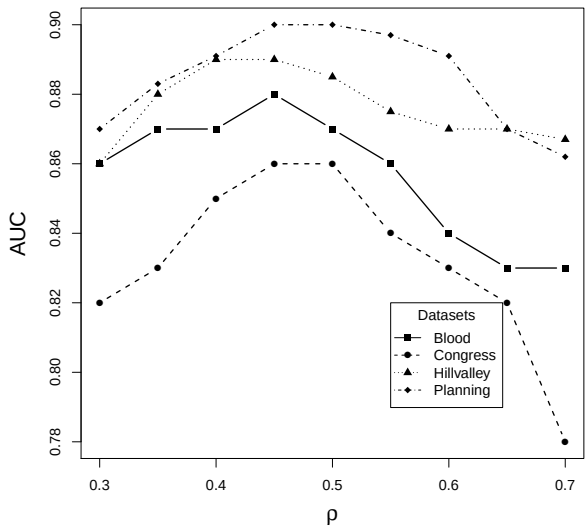


Figure 5.2: Impact of ρ on CPXC's performance

Table 5.7: Comparison on AUC of CPXC with various baseline and local classification algorithms. D: DT, L: Log, N: NBC, S: SVM. N-D: using NBC as baseline and DT as local classification algorithms resp.

Dataset	**D-D**	**D-L**	**D-N**	**L-D**	**L-L**	**L-N**	**N-D**	**N-L**	**N-N**	**S-D**	**S-L**	**S-N**
Planning	0.87	0.89	0.88	0.9	0.89	0.89	0.89	0.9	0.88	0.87	0.88	0.86
Monk 2	0.89	0.89	0.89	0.9	0.86	0.87	0.9	0.87	0.87	0.9	0.89	0.9
ILPD	0.91	0.88	0.89	0.91	0.87	0.88	0.91	0.89	0.9	0.89	0.87	0.9
Blood	0.87	0.82	0.85	0.88	0.78	0.86	0.88	0.82	0.84	0.86	0.83	0.88
Poker	0.84	0.86	0.88	0.83	0.88	0.88	0.85	0.87	0.9	0.88	0.87	0.89
HillValley	0.86	0.81	0.84	0.84	0.8	0.9	0.89	0.89	0.83	0.85	0.82	0.83
StatLog	0.9	0.87	0.9	0.9	0.86	0.9	0.91	0.88	0.9	0.9	0.88	0.89
Congress	0.8	0.85	0.87	0.83	0.75	0.84	0.86	0.86	0.74	0.88	0.85	0.89
Average	0.868	0.859	0.875	0.874	0.836	0.878	0.886	0.873	0.858	0.879	0.861	0.880
EEG	0.99	0.96	0.98	0.93	0.93	0.98	0.99	0.9	0.99	0.98	0.96	0.97
Monk1	0.98	0.98	0.98	0.98	0.96	0.97	0.98	0.97	0.97	0.98	0.97	0.98
Monk3	0.98	0.99	0.98	0.98	0.98	0.99	0.99	0.98	0.98	0.98	0.98	0.98
Climate	0.97	0.99	0.98	0.96	0.99	0.98	0.97	0.99	0.98	0.97	0.99	0.98
Banknote	1	0.99	1	1	0.99	1	1	0.99	1	1	0.99	1
Bank marketing	0.98	0.97	0.98	0.99	0.96	0.97	0.99	0.98	0.97	0.99	0.98	0.99
WholeSale	0.99	0.96	0.96	0.99	0.98	0.98	0.96	0.99	0.96	0.96	0.96	0.96
Mammography	0.97	0.96	0.98	0.97	0.94	0.97	0.98	0.96	0.97	0.97	0.97	0.97
Steel	0.98	0.99	0.97	0.99	0.97	0.98	0.99	0.98	0.98	0.99	1	0.98
Average	0.982	0.977	0.979	0.977	0.967	0.980	0.983	0.971	0.978	0.980	0.978	0.979

probabilistic loss function is the best option to use in CPXC algorithm. Probabilistic and Pearson shows similar results but the probabilistic loss function is slightly better, and both are significantly better than binary loss function. The binary function has negative impact on the CPXC's performance, since it does not make any difference between two falsely classified data instances and both are located in LE dataset. If we use binary loss function, we do not need to have ρ anymore.

Impact of objective function and local model weighting method: Figure 5.5 represents CPXC's performance when the objective function changes. The curve shows the best option is TER. One possible explanation is that TER helps CPXC to pick patterns whose local classifiers correct more errors; when the other objective functions are used, CPXC may pick patterns that have high AUC (or ACC) because the data instances in matching dataset are easy to classify, and the baseline classifier is already accurate. Figure 5.6 also represent the impact of local weighting

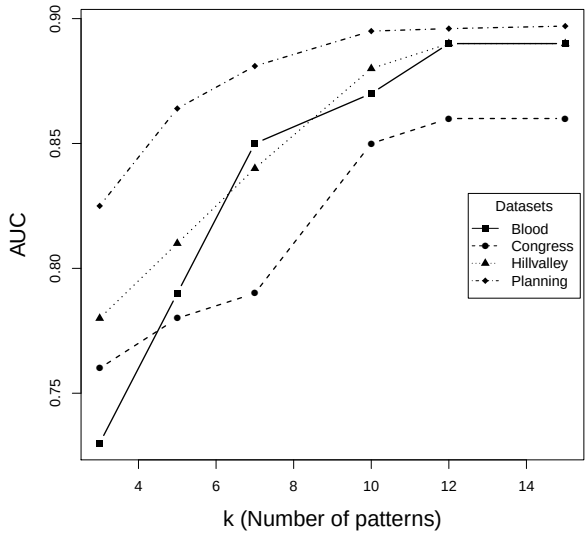


Figure 5.3: Impact of number of patterns on CPXC's performance

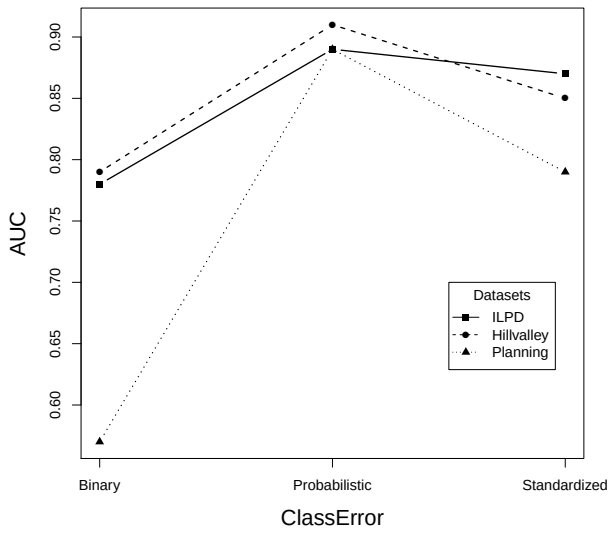


Figure 5.4: Impact of loss function on CPXC's performance

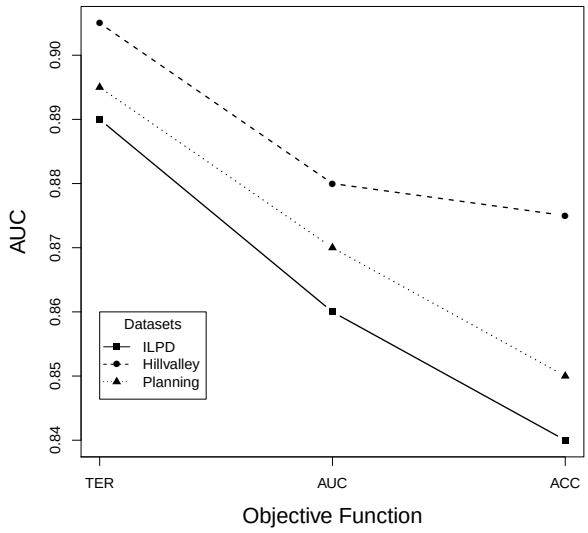


Figure 5.5: Impact of search's objective function on CPXC's performance

method on the CPXC's performance.

Impact of pattern filtering: Impact of applying pattern filtering methods on the CPXC's performance is investigated using the Blood dataset. If we do not apply filtering methods, the number of patterns is 517, AUC is 0.88, and the running time is 143 seconds. After applying the filters, the number of patterns drops to 313, AUC is still 0.88, and the running time is 88 seconds. We did not use Jaccard similarity filtering since the number of patterns was not very large.

5.3.6 Running Time and Memory Usage

Table 5.8 shows that CPXC typically uses between 0.5 and 3.7 minutes of computation time and a small amount of memory. Experiments were run on a single processor machine with a 2.8 GHz CPU and 8 GB RAM. CPXC's computation time is affected by the number of PIPs (namely contrast patterns that remain after all the filters), and the numbers of instances/attributes. Clearly CPXC computes more accurate classifier at the cost of more time when compared against other algorithms, and the amount of time used by CPXC is not too long; the extra computing time is worth it since CPXC produces significantly more accurate classifiers. Building the local classifiers is the most expensive step, taking about 80% of the total computation time.

Table 5.9 reports experiments on the scalability of CPXC. (a) For scalability w.r.t. the number of instances, we used random shuffling plus partitioning to form five datasets that respectively contain 20%, 40%, 60%, 80%, and 100% of EEG's 14980 instances. While CPXC's running time on them

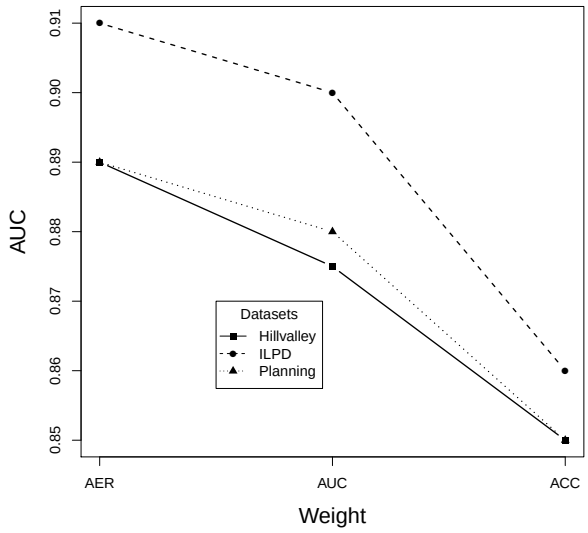


Figure 5.6: Impact of local classifier weighting methods on CPXC's performance

Table 5.8: CPXC's running time and memory usage

Dataset	**CPXC time^s**	**CPXC memory^{MG}**	**Number of PIPs**	**SVM time^s**	**Boosting time^s**	**RF time^s**	**DT time^s**
Planning	222	0.8	1115	0.12	1.11	0.29	0.08
ILPD	90	0.2	201	0.15	1.75	0.61	0.06
Monk 3	39	0.15	81	0.14	1.27	0.37	0.09

(see the EEG row) increases in an exponential manner, it can build a classifier in a fairly short time. (b) For scalability w.r.t. the number of dimensions, we used projections of HillValley (containing 100 attributes) to form five new datasets (containing respectively 20, 40, 60, 80, 100 attributes). Interestingly, CPXC's running time (see the HillValley row) is longest when the number of attributes is the smallest. This is because the bulk of CPXC's computation time is spent on building local classifiers, and many patterns are removed when the number of attributes becomes large (due to the filtering rule of "removing a pattern p if $|mds(p)| \leq \#attributes$").

Table 5.9: Scalability of CPXC (running time in minutes)

Dataset	Type	20%	40%	60%	80 %	100%
EEG	# of instances	3.8	8	17.5	33.1	63.2
HillValley	# of dimensions	11.2	9.7	6.5	4.2	2.8

5.4 Contrast Pattern Aided Classification on Imbalanced datasets

In some classification problems, the class distribution is not uniform among the classes. Often, they are composed by two classes: The majority (negative) class and the minority (positive) class. This type of problems is known as the classification on imbalanced datasets. In this section, we present the adoption of CPXC to tackle imbalanced datasets.

5.4.1 Introduction

Classification on imbalanced datasets has gained attention recently due to the growing number of applications in different areas. For example, in a dataset for online banking fraud, there are just five fraudulent transactions from approximately 300000 transactions in a day [Wei et al. 2013]. Some of the applications in medicine are detection of microcalcifications in mammogram images [N and Sheshadri 2012], nosocomial infections [Cohen et al. 2006] and, liver and pancreas disorders [Li et al. 2010].

Most of the classification algorithms are unable to produce accurate and interpretable models to classify imbalanced datasets. In such cases, traditional classifiers tend to be overwhelmed by the majority class and easily ignore the minority class; consequently, test instances belonging to the minority class are misclassified more often than those belonging to the majority class. As an example, if 99% of the instances are from negative class, it is hard for a classifier to do better than the 99% accuracy by labeling everything as the majority class. In some applications, the correct classification of minority instances are more important than the majority class. For example, in a disease diagnostic problem where the disease cases are much less than normal population and the goal is identifying patients with at high risk of developing the disease.

There are two type of methods developed to handle imbalanced classification. In the first group called cost sensitive learning methods, they try to maximize accuracy while penalizing misclassified instances of the minority class [Domingos 1999] [Pazzani et al. 1994]. In the second group called sampling methods, their approach is undersampling majority class or oversampling minority class or both [Chawla and Bowyer 2002].

In a dataset with imbalanced class distribution, the most obvious problematic characteristic is the skewed data distribution between classes. However, there are many studies discussed that the skewed data distribution is not the only factor prevents traditional classifiers from producing highly accurate models. Other important challenges include small sample size, heterogeneity of the datasets, and the existence of within-class data subgroups.

To tackle those challenges described above, we adopt Contrast Patterns Aided Classification (CPXC) to develop a new method called Contrast Pattern Aided Regression on Imbalanced Datasets (CPXCim) to handle imbalanced classification problems. The CPXCim algorithm returns a set of contrast patterns associated to a set of highly accurate, balanced, and interpretable local models. Like CPXC, CPXCim has several significant advantages including high prediction accuracy and the ability to extract subgroups data points, often outperforming state-of-the-art imbalanced classifiers by big margin. The prediction models produced by CPXCim are easy to understand as well.

CPXCim has three main contributions:

- We adopt CPXC methodology to handle imbalanced classification problems.
- We apply a weighting method to promote minority class.
- We introduce a new filtering method to remove imbalanced local models.

5.4.2 CPXCim Algorithm: New Techniques

In previous sections, we presented the main steps of CPXC. CPXCim algorithm is very similar to CPXC. Therefore, in this section, we will only introduce the novel techniques that we have used in CPXCim. Two new techniques that we used in CPXCim are a new filtering method, to identify balanced local models, and a novel weighting method, to promote instances in the minority class.

5.4.2.1 Using Weights to Promote Misclassified Minority Class Instances

Traditional classification algorithms often misclassify the instances in a minority class. This misclassification is due to the design of those algorithms, which intends to ignore the minority instances and generate a model that represents the majority of the data points. Weighting techniques often promote the minority instances by increasing the cost of misclassifying those instances.

In our experiments, we applied CPXC on imbalanced datasets, which outperformed imbalanced classification algorithms such as SMOTE and SMOTE-TL. However, we noticed the presence of few misclassified minority instances in Small Error class (SE), rather than Large Error class (LE). Therefore, we decided to apply a weighting technique to move those misclassified minority instances from SE to LE.

We used a simple technique to promote minority instances. Let's assume $err(h_b, x)$ is the error of baseline classifier h_0 on a particular instance x which is defined as $|y - h_b(x)|$ where it is assumed $y \in 0, 1$ and $h_b(x)$ is h_bs predicted probability for "xs class label is 1". The *weighted error* of baseline model h_b on instance x is defined as:

$$err^*(h_b, x) = \begin{cases} err(h_b, x) \times \delta & \text{if } x \in \textit{minority class instances} \\ err(h_b, x) & \text{if } x \in \textit{majority class instances} \end{cases} \tag{5.9}$$

where $\delta \in \{2, 4, 8, 16\}$.

5.4.2.2 A New Filtering Method to Identify Balanced Local Models

As we discussed earlier, sampling is one of the well-known methods developed for imbalanced classification problems. The goal of sampling methods including oversampling or undersampling is to make a dataset more balanced in terms of class distribution, which can be done by generating samples from the minority class or removing samples from the majority class.

In step 6 of CPXC algorithm, there are some filters designed to reduce the number of patterns and remove those patterns of low utility. Experimental results of using CPXC on imbalanced datasets revealed that many of the matching datasets associated to the contrast patterns are imbalanced. Therefore the local models built on the imbalanced matching datasets of those patterns are performing poorly. In CPXCim, we use a new filter designed to remove imbalanced matching datasets.

Definition: [Orriols-Puig and Bernadó-Mansilla 2009] The *Imbalance Ratio (IR)* of dataset D is

$$IR(mt(p, D)) = \frac{Number\ of\ instances\ in\ the\ majority\ class}{Number\ of\ instances\ in\ the\ minority\ class} \tag{5.10}$$

If the imbalance ratio of matching dataset $mt(p, D)$ is more than 3, pattern p is filtered. IR filter can help to exclude all imbalanced matching datasets.

5.4.3 Experimental Evaluations

We now use experimental results to evaluate the performance of CPXCim and the usefulness of various technical ideas proposed in this part. Our experimental study uses 10 benchmark datasets from [Loyola-González and Al. 2016] and compares CPXCim with classification algorithms developed for imbalanced dataset including SMOTE [Chawla and Bowyer 2002] and SMOTE-TL [Batista et al. 2004].

Table 5.10: Accuracy of CPXCim, SMOTE, SMOTE-TL and the best reported by [Gonzalez, 2016]

Dataset	Number of instances	Number of variables	Imbalance ratio	CPXCim	SMOTE	SMOTE-TL	Best of other
yeast0256vs3789	1004	8	9.14	0.942	0.7728	0.772	0.799
led7digit02456789vs1	443	7	10.97	0.978	0.8919	0.897	0.906
flareF	1066	11	23.79	0.883	0.7463	0.809	0.827
winequalityred4	1599	11	29.17	0.76	0.6008	0.59	0.669
pima	768	8	1.87	0.847	0.7407	0.744	0.754
abalone17vs78910	2338	8	39.31	0.977	0.7645	0.776	0.8
cleveland0vs4	177	13	12.62	0.963	0.801	0.794	0.866
pageblock0	5472	10	8.79	0.98	0.9448	0.945	0.957
vehicle1	846	18	2.9	0.879	0.7531	0.766	0.778
wisconsin	683	9	1.86	0.991	0.9666	0.979	0.979
Average				**0.92**	0.798	0.807	0.833

5.4.3.1 Datasets and Experiment Settings

We used the following datasets from [Loyola-González and Al. 2016]: yeast0256vs3789, led7digit02456789vs1, flareF, winequalityred4, pima, abalone17vs78910, abalone17vs78910, pageblock0, vehicle1, wisconsin. Our criteria to choose datasets are the number of instances, the number of variables and the imbalance ratio. The imbalance ratio varies from 1.87 in pima to 39.31 in abalone17vs78910 which is extremely imbalanced. CPXCim has three parameters: *minSup*, ρ and δ. *minSup* were set at 0.02.

In our experiments, we observed that the performance of CPXCim is sensitive to the value of ρ and δ. To find the best pair of ρ and δ, we used a pool of (ρ,δ) vectors where ρ is chosen from $\{0.45, 0.5, 0.55, 0.6.\}$ and δ is chosen from $\{2, 4, 8, 16\}$. We also used 5-fold cross-validation to tune the pair of ρ and δ.

5.4.3.2 CPXCim vs Other Algorithms: Prediction Accuracy

Table 5.10 presents the AUC of CPXCim, SMOTE, and SMOTE-TL [1] and the best AUC reported by [Loyola-González and Al. 2016]. The results point out interesting findings. First, the average AUC of CPXCim on 10 benchmark datasets is 0.92 which is 14% and 15.2% more than the AUC of SMOTE and SMOTE-TL, respectively. Second, on average, CPXCim's AUC is 10.4% more than the best AUC reported by [Loyola-González and Al. 2016]. Third, the performance of CPXCim is always better than other imbalanced classifiers on these 10 datasets. So CPXCim builds significantly

[1] CAEP [Dong et al. 1999] is used to build the classifiers after applying SMOTE and SMOTE-TL

more accurate classifiers on imbalanced datasets that are challenging to traditional classification algorithms.

5.5 Conclusion

This chapter introduced (a) a new type of classifiers, namely Pattern Aided Classifiers (PXC), and (b) a new classification algorithm, namely Contrast Pattern Aided Classification (CPXC), for building accurate and interpretable PXCs. PXCs are especially suitable for applications where the underlying dataset contains highly heterogeneous subgroups whose best-fit local specific classifiers are highly different. The success of CPXC for many applications (including those that are highly challenging to existing classification methods) can be attributed to its utilization of pattern-based opportunity-guided boosting and the modeling power of pattern aided classifiers. Our experiments indicate that high heterogeneity of the underlying datasets is likely one main reason why certain applications are highly challenging to traditional classification algorithms.

6

Applications of CPXR and CPXC

CPXR, CPXC and, CPXCim are applicable to any classification and prediction problems in different fields such as medicine, economy and, environmental science. In this section, we discuss some of the applications of our proposed methodologies. In the first application, we used CPXR(Log) (CPXR(Log) is same as CPXC(Log-Log)) to predict patient's outcome within 6 months after Traumatic Brain Injury (TBI). In the second application, a new heart failure risk model is developed using CPXC algorithm. In the third application, we used CPXR to build a set of numerical models to predict saturated hydraulic conductivity (SHC) and soil water retention curve (SWRC).

6.1 Application of CPXC in Traumatic Brain Injury (TBI)

6.1.1 Introduction

Prognostic models are central to medicine; they are usually used to predict patients' outcome and patient's response to medication. Physicians routinely make their decisions on the patient treatment plan, screening, and ordering of tests and procedures, based on the prognosis or likelihood of a disease [Steyerberg et al. 2008]. Prognosis models also help on the understanding of diseases, including identifying discriminating variables highly correlated with the outcome. Medicine is moving from a traditional subjective one to an evidence-based one, which uses prediction models built from population samples to inform clinical decision-making [Trisha 2014].

Traumatic brain injury (TBI) is an important public health problem and a leading cause of death and disability worldwide: Every year, more than 1.5 million people die and hundred of millions need emergency treatment [Perel et al. 2006]. In the US, CDC estimated that 2.4 million emergency room visits, hospitalizations, and deaths are related to TBI and $76.5 billion dollars including direct and indirect costs (excluding combat related treatments) in 2010 [CDC]. While confident predictions

could usually be made 24 hours after the injury, they are hard to make at admission time [Jennett et al. 1976]. Physicians need to make vital decisions ranging from whether to perform/withdraw certain treatments based on their prognosis evaluation [Perel et al. 2007], and they need accurate prognostic models that only use admission time data to make time-critical clinical decisions.

Challenges in clinical modeling include the following five (some are for general predictive clinical modeling, and some are for prognostic modeling for traumatic brain injury).

- Accuracy of prediction models is the most important aspect for clinical prediction modeling, as making a wrong decision in medicine may put a human's life in danger.
- Prediction models for medicine should be easy to interpret, so that physicians can (i) explain critical medical decisions to patients and their families and (ii) can identify the important risk factors for the disease under consideration.
- Prediction models for medicine should avoid overfitting as much as possible so that they can be used to make accurate predictions on new cases.
- Prediction models should allow physicians to make early decisions. This is often critical, as an early decisions will allow hospitals to make early effort on patients who will likely benefit from the treatment. For traumatic brain injury, correct treatment decisions made at the time of admission, with admission time data, will help the patient to recover better while delayed decisions will diminish their chance of recovery.
- As will be discussed in the Related Work section, TBI patients in different population groups require different prediction models. In fact, as will be shown in this study, this heterogeneity is not limited to known population groups; TBI is an illness having diverse predictor-response variable relationships[1] [Dong and Taslimitehrani 2015].

For prognostic modeling on TBI using admission time data, the models produced by CPXR(Log) achieved AUC as high as 0.93 and specificity as high as 0.97, much better than those reported by previous studies. Each prediction model produced by CPXR(Log) contains several interpretable local prediction models for different patient groups, indicating that there are several different kinds of patients that should be evaluated differently for TBI outcome prediction. We present a complete CPXR(Log) prediction model, containing the patterns and the local logistic regression models, for the Unfavorable dichotomized version of GOS using 15 predictor variables. We also study the odds

[1] When we say "an illness has diverse predictor-response relationships", we mean the data associated with the illness contains multiple logical data groups whose fitted regression models are highly different. Illness can be other things.

ratio differences of the predictor variables based on different logistic regression models; we provide predictor variables whose odds ratios in some local model (of the CPXR(Log) model) differ from that in the global model significantly (including variables whose odds ratios change by more than 6 folds). The example CPXR(Log) model demonstrates that CPXR(Log) can also extract informative multi-variable outcome-related interactions among subsets of variables, which are hard to identify by standard logistic regression when the number of variables is large.

6.1.2 TBI's Related Work

The related works belong to two main groups. (a) Studies on general clinical prediction models: Clinical prediction modeling is a very broad and active area of research. Most recent articles on clinical prediction modeling used Logistic Regression (e.g. [Bagley et al. 2001]), while others used methods such as Decision Trees (e.g. [Brown et al. 2005]), Random Forest (RF) (e.g. [Kennedy et al.]) and Support Vector Machine (SVM) (e.g. [Yu et al. 2010]);

(b) Studies on TBI related prediction models: Many studies have been reported on prediction modeling for predicting the outcome after TBI event. A preeminent study is IMPACT (International Mission for Prognosis and Analysis of Clinical Trials in TBI) [IMPACT], which collected data for nearly 10 years and developed and validated many prognostic models for classification and prognostic risk calculation. In [Murray et al. 2007], Murray et al. examined important risk factors on TBI patients' outcome based on a cohort of 8686 patients from multiple clinical trials. They fitted a proportional odds model and found age, GSC score, pupil response and CT characteristics are the most powerful prognostic risk factors. Hukkelhoven et al. [Hukkelhoven et al. 2003] performed a study to detect critical age threshold on TBI patients' outcome on a set of 5600 patients. CRASH (Corticosteroid Randomization After Significant Head Injury) [CRASH Trial Collaborators 2005] is another major trial on TBI which ended up with 10008 patients; it also developed prognostic models and risk calculators. Reference [Steyerberg et al. 2008] built prognostic models to predict Mortality and Unfavorable, where the outcome classes are determined based on the GOS score at 6 months after the surgery, using logistic regression. Reference [Brown et al. 2005] developed similar models using decision tree analysis. Reference [Maas et al. 2007] found that heterogeneity of head injuries is a challenge in TBI prognostic models; CRASH [Perel et al. 2008] found that prognostic models for TBI patients' outcome for low, middle and high-income countries differ significantly, which is in agreement with the heterogeneity findings of [Maas et al. 2007].

6.1.3 Results and Discussion

This section presents the results of CPXR(Log) on prediction of 6-month outcome after moderate or severe TBI. It has two focuses, first on accuracy of CPXR(Log) models, and second on new insights on TBI offered by CPXR(Log) models (which could be useful to medical scientists and physicians).

- For the former, we mostly compare CPXR(Log) against standard logistic regression (denoted by SLogR), while briefly comparing against state-of-the-art classification algorithms such as SVM and RF. The results indicate that CPXR(Log) is more accurate, outperforming SLogR and others significantly.

- For the latter, the CPXR(Log) models present new patterns that capture outcome-related interactions among variables and that define groups of patients whose outcome should be predicted using their own local prediction models instead of the global logistic regression model. Moreover, based on CPXR(Log), we present variables having high odds ratios for certain patient groups (defined by patterns used by the CPXR(Log) models) and having low odds ratio based on the SLogR model.

We believe, important variables for TBI include those whose CPXR(Log) based odds ratios differ from their SLogR based odds ratio by large margins, and those that occur in patterns used by the CPXR(Log) models. Regarding CPXR(Log)'s parameters, we used fixed values $minSup = 0.02$ and $\rho = 0.4$ CPXR(Log), and we used default settings of the R [Team 2014] packages for SLogR, SVM and RF.

6.1.3.1 TBI Dataset

The TBI dataset considered in this study[2] is from [Steyerberg et al. 2008], which will be called TBI in this paper, on patients from an International and US Tirilazad trials. It contains 2159 instances and 15 predictor variables; its missing predictor variable values were treated using multiple imputations as suggested by [Steyerberg 2008].

The outcome variable of TBI is assessed with the Glasgow outcome scale (GOS), which has been widely used in brain injury studies. The scale ranges from dead (GOS 1), vegetative state (GOS 2), severe disability (GOS 3), moderate disability (GOS 4), to good recovery (GOS 5). The predictor variables belong to three groups:

- **Basic variables (4 variables):** cause of injury, age of patient, GCS motor score, and pupil reactivity.

[2]Datasets on traumatic brain injury are not publicly available and hence we are limited to this dataset in this paper.

- **Computed-tomography variables (7 variables):** hypoxia, hypotension, CT characteristics (Marshall CT classification), traumatic subarachnoid hemorrhage (tSAH), epidural hematoma (EDH), compressed cistern at CT, and midline shift more than 5mm.
- **Lab variables (4 variables):** glucose, pH, sodium and Hb (hemoglobin)

Details on these predictor variables can be found in [Perel et al. 2008], [Steyerberg et al. 2008].

Prognostic models studied before consist of all six prognostic models for the six combinations of two dichotomized versions of GOS and three subsets of variables. These combinations were examined in previous studies [Perel et al. 2008], [Steyerberg et al. 2008] on TBI, allowing us to compare the performance of our method against [Perel et al. 2008], [Steyerberg et al. 2008].

The two dichotomized versions of GOS are: mortality outcome (versus survival) and unfavorable outcome (versus favorable). For the first, called "Mortality", all cases with the dead outcome (GOS 1) belong to the "mortal" class and all others (GOS 2–5) are in the "survival" class. For the second, called "Unfavorable", all cases with dead, vegetative and severe disability outcomes (GOS 1–3) are in the "unfavorable" class, whereas all cases with moderate disability and good recovery outcomes (GOS 4–5) are in the "favorable" class. The three variable sets considered are: Basic, Basic+CT, and Basic+CT+Lab, consisting 4, 11, 15 variables respectively [3].

6.1.3.2 Evaluation on Prognostic Model Accuracy Measures

We now compare CPXR(Log) against SLogR on several measures concerning model accuracy. The results show that CPXR(Log) outperforms SLogR consistently and by big margins. The strong outperformance implies that TBI has diverse predictor-response relationships.

Tables 6.1 and 6.2 present the performance of prognostic models built by SLogR and CPXR(Log). Figure 6.1 shows the ROC curves of all six models developed by SLogR and CPXR(Log); solid lines represent curves of SLogR and dashed lines represent curves for CPXR(Log). (Figure 6.2 compares ROC curves of CPXR(Log) against that of SVM and RF.)

The results reported here agree with those reported in [Steyerberg et al. 2008]. Specificity, sensitivity, and accuracy are for the classifier using the logistic regression models with 0.5 as cutoff: If the predicted value is larger than 0.5 then the predicted class is 1, otherwise the predicted class is 0. Sensitivity (aka the true positive rate, or recall) is the proportion of actual positives which are correctly identified as such. Specificity (aka the true negative rate) is the proportion of negatives which are correctly identified as such.

[3] We add qualifiers to the names of models to avoid confusion. Specifically, we will use Method-DichotomizedName-VariableSet as model names. For example, the SLogR-Mortality-(Basic+CT) model refers to the model built by SLogR for Mortality using the Basic+CT variables.

Table 6.1: SLogR performance on accuracy

Model		**Basic**	**Basic+CT**	**Basic+CT+Lab**
Mortality	Specificity	0.95	0.95	0.94
	Sensitivity	0.18	0.32	0.36
	Accuracy	0.77	0.8	0.8
	F_1	0.27	0.42	0.46
	AUC	0.72	0.78	0.8
	χ^2	2192	2183	2094
Unfavorable	Specificity	0.85	0.85	0.84
	Sensitivity	0.52	0.6	0.61
	Accuracy	0.72	0.75	0.75
	F_1	0.59	0.66	0.66
	AUC	0.76	0.8	0.81
	χ^2	2174	2172	2137

Apparently, all six CPXR(Log) models outperformes corresponding SLogR models on all performance measures. In particular, AUC of all CPXR(Log) models improves that of SLogR models by 11.7% on average (all six models), and the improvement is 12.9% for Mortality models and 10.4% for Unfavorable models. Moreover, on average over all six models, specificity, sensitivity, accuracy, and χ^2 of the Basic+CT+Lab models built by CPXR(Log) improve over those of SLogR by 8%, 18%, 16% and 28% respectively. Interestingly, CPXR(Log) achieves more improvement over SLogR on χ^2 concerning Mortality models than Unfavorable models: The average improvement for Mortality models is 29.4% and, it is 26.3% for Unfavorable models.

One main strength of CPXR(Log) is its ability to effectively utilize more variables to derive more accurate models, which is similar to CPXR for linear regression [Dong and Taslimitehrani 2015]. Indeed, while both CPXR(Log) and SLogR obtained improvement on AUC when more variables are used, CPXR(Log) obtained larger improvement in all cases (see Table 6.3). Moreover, when more variables are used, CPXR(Log) achieved larger improvement on AUC over SLogR, as shown in Table 6.4. CPXR can also effectively extract useful information capturing interactions among multiple predictor variables that are often missed by traditional regression and classification methods.

Interestingly, Figures 6.1 and 6.2 show that the ROC curves of CPXR(Log) always have larger

Table 6.2: CPXR(Log) performance on accuracy

Model		Basic	Basic+CT	Basic+CT+Lab
Mortality	Specificity	0.96	0.96	0.97
	Sensitivity	0.18	0.42	0.46
	Accuracy	0.78	0.85	0.89
	F_1	0.28	0.53	0.58
	AUC	0.8	0.88	0.92
	χ^2	1801	1483	1290
Unfavorable	Specificity	0.89	0.87	0.91
	Sensitivity	0.54	0.65	0.72
	Accuracy	0.75	0.79	0.87
	F_1	0.63	0.7	0.76
	AUC	0.82	0.87	0.93
	χ^2	1848	1601	1327

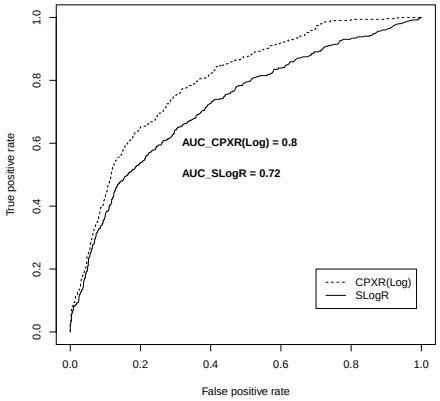
true positive rate for every false positive rate, than that of SLogR, SVM, and RF. Another observation is that for both SLogR and CPXR(Log), prognostic models for Mortality are more accurate than those for Unfavorable, suggesting that the unfavorable class is harder to model.

6.1.3.3 Overfitting

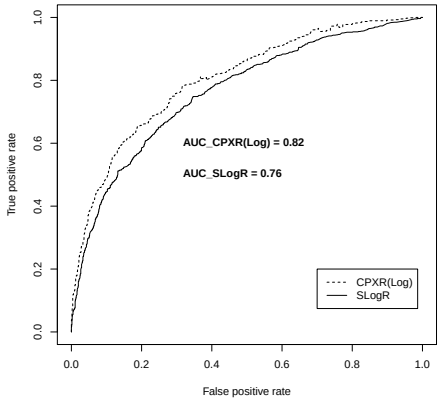
Overfitting is a major issue in clinical prediction modeling; it happens when a prediction model (or classifier) is much more accurate on training data than on test data that are unknown to the model. Overfitting models are not desirable, as end users including physicians cannot be very confident in using them to make predictions on new cases. When comparing prediction models, more accurate models are preferred; among equally accurate models, the less overfitting ones are preferred.

Table 6.5 gives patterns of the CPXR(Log)-Unfavorable-(Basic+CT+Lab) model, respectively. Table 6.6 compares odds ratios of variables in the SLogR and CPXR(Log) models. Odds ratio (OR) is a popular measure to quantify how strongly the level of a predictor variable x_i is associated to the response variable [CORNFIELD 1951]. The odds ratio of x_i is often estimated from a logistic regression model as $OR(x_i) = e^{\beta_i}$, where β_i is the coefficient of x_i in the logistic regression model.

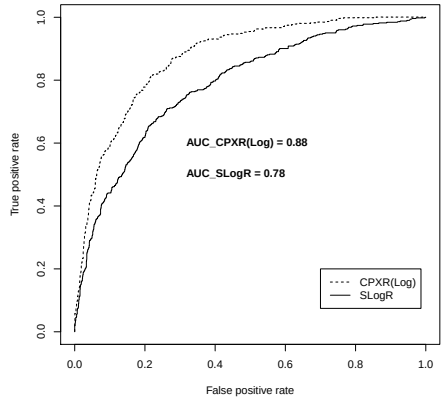
In logistic regression, we need to convert categorical variables into dummy variables. To avoid redundancy, one of the values of each categorical variable should be omitted ("treated as reference



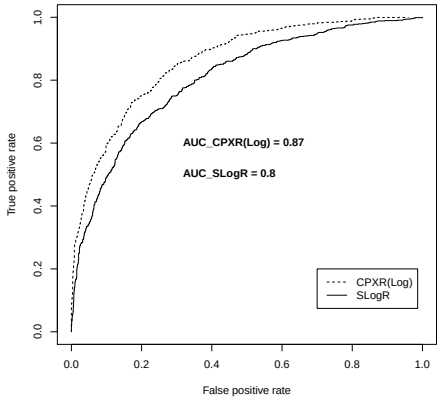
(a) Mortality-Basic models



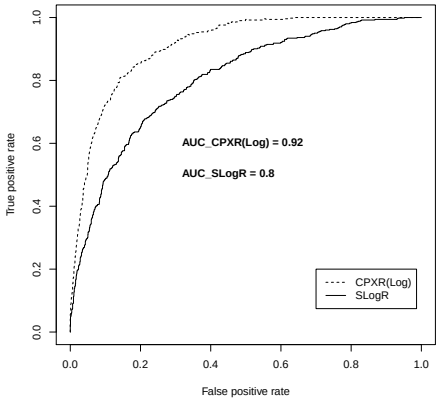
(b) Unfavorable-Basic models



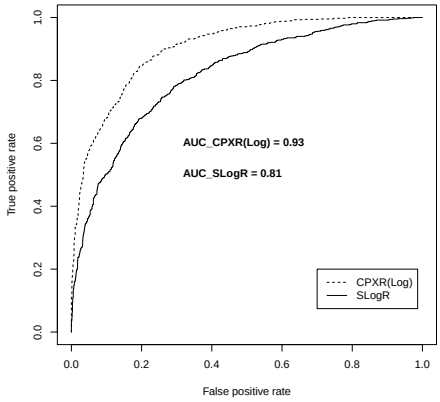
(c) Mortality-(Basic+CT) models



(d) Unfavorable-(Basic+CT) models



(e) Mortality-(Basic+CT+Lab) models



(f) Unfavorable-(Basic+CT+Lab) models

Figure 6.1: Comparison of CPXR(Log) and SLogR: ROC curves and AUC

Table 6.3: AUC improvement when more variables are used by CPXR(Log) and SLogR

	Mortality		Unfavorable	
Variable set change	**CPXR(Log)**	**SLogR**	**CPXR(Log)**	**SLogR**
Basic → Basic+CT	10%	7.7%	6%	5.2%
Basic+CT → Basic+CT+Lab	4.5%	2.5%	6.8%	1.25%
Basic → Basic+CT+Lab	15.0%	11.1%	13.4%	6.6%

Table 6.4: AUC improvement by CPXR(Log) over SLogR for given variable sets

Mortality			Unfavorable		
Basic	**Basic+CT**	**Basic+CT+Lab**	**Basic**	**Basic+CT**	**Basic+CT+Lab**
11.1%	12.8%	15%	7.9%	8.8%	14.8%

category"). We chose the most common value of each categorical variable as the reference category. A categorical variable for a local regression model becomes constant if the variable occurs in the pattern of the local model and is hence a constant. Reference categories and categorical variables involved in the patterns are both specified as "ref" in Table 6.6.

Large differences in odds ratio can be of interest to physicians, as they indicate for certain large population groups, risk should be evaluated in a manner different from how risk is evaluated based on the SLogR model. Large difference can be for cases where odds ratio in CPXR(Log) models is significantly higher or lower than that in SLogR models.

There are quite a number of variable and value pairs where odds ratio differences are large. In Table 6.6, we use the bold font to indicate such pairs where the odds ratio in the CPXR(Log) model is at least twice of that in the SLogR model, and we use the italic font to indicate cases where the odds ratio in the CPXR(Log) model is at most half of that in the SLogR model. We use underline to indicate some other cases where the odds ratio in the CPXR(Log) model is much larger than that in the SLogR model although not at least twice as much. To save space, we omit rows having no large differences.

The largest odds ratio difference is 6.40 fold, for $6.79 < pH \leq 7.67$, whose odds ratio is 0.84 according to SLogR model and it is **5.38** for Model I of CPXR(Log). The largest odds ratio difference in absolute value is 7.07, for $reactivity =$ "$No\ reactive$", whose odds ratio is 2.66 according to SLogR model and it is **9.73** for Model II of CPXR(Log). The largest odds ratio decrease is 16.8 fold, for

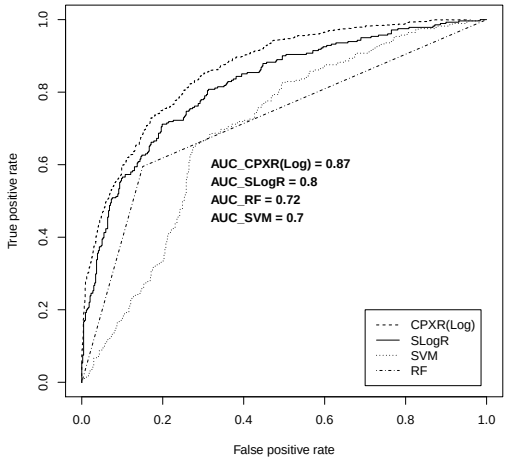


Figure 6.2: ROC curves and AUC of CPXR(Log), SLogR, SVM, and RF on test data of models built from training data

$6.79 \leq PH < 7.67$ whose odds ratio is 0.84 according to SLogR model and it is **0.05** for local Model VI of CPXR(Log).

An example where the CPXR(Log) model corrected a large prediction error: Consider this 15 years old patient with TBI due to a motorbike accident, who has the following characteristics at the admission time:

> GCS motor score = 5. No reactive pupil. No hypoxia. No hypotension. CT scan = V. (high lesion > 25 mm and not surgically evacuated.) No tSAH. Has epidural hematoma. Has fully compressed cisterns. Has midline shift. Glucose = 9.06 mmol/l. PH = 7.37. Sodium = 141 mmol/l. Hb = 14.4 g/dl.

Since the patient is young, having no evidence of hypoxia, hypotension, and tSAH, the baseline regression model built by SLogR predicts that the patient's probability of survival is 0.78. However the observed outcome (after 6 months) is death. In contrast, since this patient matches pattern II of the CPXR(Log) model, i.e., "(CT classification = V) AND (midline shift more than 5mm) AND ($0.56 <$ glucose ≤ 10.4)" is true, the CPXR(Log) model more accurately predicts probability of survival to be just 0.31, using local model II in Table 6.6 associated with this pattern ("CT scan = V" means "high lesion > 25 mm and not surgically evacuated."). Incidentally, in our analysis, we found that the SLogR model made many big prediction errors in young patients.

Table 6.5: Pattern, arr, coverage of local models of CPXR(Log)-Unfavorable-(Basic+CT+Lab) model

Patterns	**arr**	**Coverage**	**Model**
(CT classification = III)	15%	20%	I
(CT classification = V) AND (midline shift more than 5mm) AND ($0.56 < \text{glucose} \leq 10.4$)	12%	15%	II
(No compressed cistern) AND (No midline shift more than 5mm) AND ($7.22 < \text{PH} \leq 7.45$)	10%	40%	III
($10.77 < \text{glucose} \leq 21.98$)AND ($134 < \text{sodium} \leq 144$)	18%	18%	VI
(No Hypotension)AND ($134 < \text{sodium} \leq 144$) AND ($10.55 < \text{HB} \leq 14.57$) AND (With tSAH)	19%	20%	V
(No tSAH) AND ($134 < \text{sodium} \leq 144$) AND ($10.77 < \text{glucose} \leq 21.98$) AND (No Hypotension) AND (No midline shift) AND (One reactive pupil)	19%	20%	VI
(No tSAH) AND (One reactive pupil)	18%	40%	VII

6.1.3.4 Conclusion

We provided an effective new method, CPXR(Log) for logistic regression and clinical predictive modeling. CPXR(Log) achieved much higher accuracy than standard logistic regression on traumatic brain injury (TBI). We also presented our CPXR(Log) model on TBI on admission time data, including patterns and local models, and presented new odds ratios of predictor variables based on CPXR(Log), including those whose odds ratio are highly different from the SLogR model based odds ratios. We hope that these findings will have significant value in accurate clinical prognostic decision-making, including on TBI. In general, CPXR(Log) can effectively handle data with diverse predictor-response relationships.

Table 6.6: Odds ratios of predictor variables in the SLogR and CPXR(Log) models

Variables	Coding	Odds Ratios (95% CI)							
		SLogR	Model I	Model II	Model III	Model IV	Model V	Model VI	Model VII
Cause	Motorbike	0.87 (0.75-1.0)	0.85 (0.73-1.0)	**2.19** (1.8-2.6)	0.75 (0.65-0.85)	*0.36* (0.3-0.4)	0.5 (0.4-0.6)	1.08 (0.9-1.3)	1.2 (1.0-1.4)
	Assault	1.07 (0.9-1.2)	**2.86** (2.5-3.4)	0.94 (0.8-1.1)	0.69 (0.6-0.8)	0.62 (0.55-0.75)	*0.3* (0.25-0.35)	**2.94** (2.65-3.2)	**3.5** (3.0-4.0)
	Other	1.37 (1.1-1.6)	1.64 (1.5-1.8)	1.83 (1.6-2.1)	1.02 (0.8-1.2)	1.21 (1.0-1.4)	*0.47* (0.4-0.55)	1.72 (1.5-1.9)	1.35 (1.2-1.5)
Motor score	IV	0.37 (0.3-0.5)	*0.16* (0.1-0.2)	**0.77** (0.65-0.8)	0.47 (0.4-0.55)	0.32 (0.25-0.4)	0.31 (0.25-0.35)	0.48 (0.4-0.6)	0.58 (0.5-0.7)
	V/VI	0.23 (0.2-0.3)	*0.12* (0.1-0.15)	**0.55** (0.5-0.6)	0.22 (0.2-0.25)	0.18 (0.15-0.2)	*0.1* (0.05-0.15)	0.4 (0.35-0.45)	**0.46** (0.4-0.5)
Pupillary reactivity	No reactive	2.66 (2.3-3.0)	1.7 (1.5-1.9)	**9.73** (8.0-11.0)	2.37 (2.0-2.8)	1.69 (1.4-2.0)	2.87 (2.5-3.3)	1.0 (ref)	1.0 (ref)
Hypoxia	Yes	1.64 (1.45-1.8)	1.32 (1.1-1.5)	1.35 (1.2-1.5)	1.59 (1.4-1.8)	1.58 (1.4-1.8)	1.41 (1.2-1.6)	**3.18** (2.8-3.6)	1.58 (1.4-1.8)
Hypotens	Yes	1.19 (1.0-1.4)	2.25 (1.9-2.5)	1.0 (ref)	1.08 (0.9-1.3)	**2.44** (2.1-2.7)	1.0 (ref)	1.0 (ref)	1.18 (1.0-1.4)
CT classification	II	2.35 (2.0-2.7)	1.0 (ref)	1.0 (ref)	1.87 (1.6-2.2)	1.3 (1.1-1.5)	*0.33* (0.3-0.4)	1.79 (1.6-2.0)	2.23 (2.0-2.5)
	III	3.99 (3.5-4.5)	1.0 (ref)	1.0 (ref)	3.63 (3.1-4.1)	2.0 (1.8-2.2)	*0.6* (0.5-0.7)	7.11 (6.0-8.0)	4.32 (3.7-4.9)
	IV	3.74 (3.0-4.4)	1.0 (ref)	1.0 (ref)	5.05 (4.5-5.5)	*1.18* (1.0-1.4)	*0.71* (0.6-0.8)	3.7 (3.3-4.1)	*1.1* (0.9-1.3)
	V	4.72 (4.0-5.4)	1.0 (ref)	1.0 (ref)	2.79 (2.4-3.2)	*1.68* (1.5-1.9)	*0.68* (0.6-0.8)	6.6 (5.6-7.6)	4.02 (3.5-4.5)
	VI	5.04 (4.0-6.0)	1.0 (ref)	1.0 (ref)	3.94 (3.5-4.3)	*2.24* (2.0-2.4)	*0.8* (0.7-0.9)	4.63 (4.0-5.2)	4.76 (4.1-5.3)
Cisterns compression	Slightly	1.03 (0.9-1.1)	0.82 (0.75-0.9)	1.34 (1.2-1.5)	1.0 (ref)	0.86 (0.8-0.9)	*0.4* (0.35-0.45)	1.65 (1.45-1.85)	*0.57* (0.5-0.6)
	Fully	2.05 (1.7-2.3)	1.89 (1.7-2.1)	3.43 (3.0-4.0)	1.0 (ref)	**4.38** (3.8-5.0)	*0.75* (0.6-0.9)	2.53 (2.2-2.8)	2.09 (1.9-2.3)
Shift	Yes	1.03 (0.9-1.2)	1.18 (1.0-1.4)	1.04 (0.9-1.2)	1.0 (ref)	1.51 (1.3-1.7)	**2.49** (2.2-2.8)	1.0 (ref)	1.29 (1.1-1.5)
pH	6.79-7.67	0.84 (0.75-0.95)	**5.38** (5.0-5.8)	**4.45** (4.1-4.8)	*0.4* (0.35-0.45)	1.32 (1.2-1.45)	0.72 (0.6-0.8)	*0.05* (0.04-0.06)	*0.18* (0.15-0.21)

6.2 Application of CPXC in Heart Failure Survival Prediction Models

6.2.1 Introduction

Heart Failure (HF) is a major health issue and is one of the most common causes of hospitalization in the United States (US) with an estimated 6.6 million US adult cases in 2010 at a cost of 34.4 billion US dollars in healthcare expenses [CDC]. Identification of cost-effective strategies to reduce the incidence of hospitalization, a major driver of costs, is a major objective. Central to the management of HF is multifaceted pharmacological intervention that involves treatment of volume overload for symptom relief and disease modification in high risk patients to reduce mortality. Identification of cost-effective strategies to reduce the incidence of hospitalization, a major driver of costs, is a

major objective. Central to the management of HF is multifaceted pharmacological intervention that involves treatment of volume overload for symptom relief and disease modification in high risk patients to reduce mortality.

Accurate HF survival prediction models can be beneficial to both patients and physicians. Physicians could prescribe more aggressive treatment plans for high risk patients based on accurate risk predictions, and patients can have confidence in the treatment plan prescribed by physicians, and hence are more likely to comply with treatment [Panahiazar et al. 2015b]. However, there are many challenges, at least from the modeling perspective, in developing purely EHR-driven risk prediction models. Some of these challenges include: (1) models need to be highly accurate with very few false positive cases [Panahiazar et al. 2015a]; (2) models need to be highly interpretable [Letham et al. 2013] so that healthcare providers can apply them to identify clinically relevant prognostic markers, that allow them to make informed clinical decisions, and (3) the models need to minimize overfitting so that they are generalizable and can make accurate predictions on new cases.

To address these challenges, in this study, we apply CPXR(Log) method (Contrast Pattern Aided Logistic Regression) on HF survival prediction with the probabilistic loss function. CPXR(Log) can effectively identify important disease subgroups from patients EHR data, and it can produce localized prediction models for personalized considerations for those subgroups. The major contributions of this work include:

- We demonstrate that CPXR(Log) is a powerful methodology for clinical prediction modeling for high dimensional complex medical data: It can
  - produce highly accurate models (One CPXR(Log) model achieved an AUC and accuracy of 0.94 and 0.91, respectively, significantly outperforming models reported in prior studies).
  - help to identify and correct significant systematic errors of logistic regression models.
- We present classification models for HF which are much more accurate than logistic models and models produced by other state-of-the-art classifiers.
- Our CPXR(Log) models for HF reveal that HF is highly heterogeneous, suggesting that patients with heterogeneous characteristics (e.g, clincial characteristics, co-morbidities) should be evaluated for different HF management strategies.
- We propose a novel probabilistic loss function in the CPXR(Log) algorithm. It returns more accurate models comparing to CPXR(Log) introduced in our previous studies.

6.2.2 Study Population

Our primary goal in this study is to develop classifiers to predict survival in 1-, 2- and 5- years after HF diagnosis. Our classifiers are built using EHR data on 119,749 patients admitted to Mayo Clinic between 1993 and 2013. Some patient records (N=842) were excluded due to incomplete and missing data. In consultation with cardiologists and cardiovascular epidemiologists, the following cohort identification criteria were developed:

- A diagnosis of HF based on the ICD9-CM code (428.x).
- An EF measurement of 50% within two months of HF diagnoses.
- No prior diagnosis of coronary artery disease, myocarditis, infiltrative cardiomyopathy and severe valvular disease.
- Authorization to access EHR data for research.

Table 6.7: Demographics, vitals and lab characteristics of patients in our cohort

Age (in years)	78±10
Sex (male)	52%
Race (white)	94%
Ethnicity (Not Hispanic or Latino)	84%
BMI	28.7±11.25
Systolic Blood Pressure (mm/Hg)	120±25
Ejection Fraction	36%±10.3
Hemoglobin (g/dL)	11.8±1.2
Cholesterol (mg/dL)	144±35
Sodium (mEq/L)	128±4.2
Lymphocytes (x10(9)/L)	1.32±0.7

To be included in this cohort, patients needed to meet all four criteria, leading to a final cohort size of 5044 HF patients admitted to Mayo Clinic between 1993 and 2013. To select predictor variables, we followed the SHFM [Ouwerkerk et al. 2014] and added a series of new variables derived from the EHR data that were grouped into the following categories:

- Demographics including age, gender, race, and ethnicity.
- Vitals including Blood Pressure (BP), and Body Mass Index (BMI).

- Lab results including cholesterol, sodium, hemoglobin, lymphocytes, and ejection fraction (EF) measurements.
- Medications including Angiotensin Converting Enzyme (ACE) inhibitors, Angiotensin Receptor Blockers (ARBs), β-adrenoceptor antagonists (β-blockers), Statins, Calcium Channel Blocker (CCB), Diuretics, Allopurinol, and Aldosterone blocker.
- A list of 24 major chronic conditions [OPH] as co-morbidities.

Since our EHR data is time dependent, we considered the records that are closest to the HF event. Our class variable (response) is mortality status. For the 1-year version of the dataset, if a patient was dead within 1-year after the heart failure event, the class variable is 1, otherwise it is 0. We created 3-year and 5-year versions of the dataset similarly.

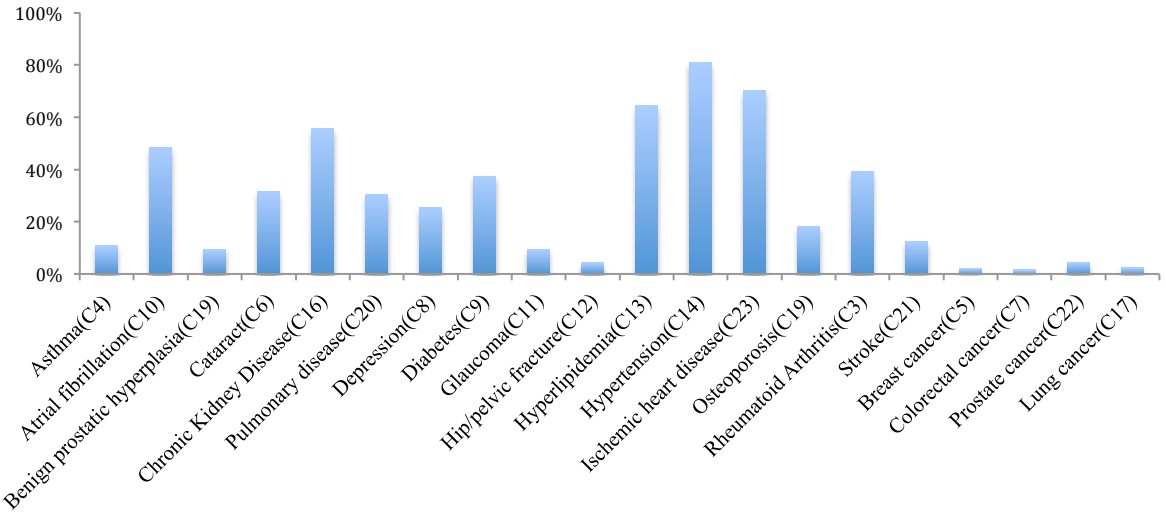


Figure 6.3: Frequency of co-morbidities in the cohort

Table 6.7 represents demographics, vitals and lab characteristics of patients in our cohort, and Figure 6.3 shows frequencies of co-morbidities. It can be observed that hypertension, ischemic heart disease, hyperlipidemia, chronic kidney disease and atrial fibrillation are the most frequent co-morbidities. Table 6.8 represents the frequency of different medication classes used in the cohort; apparently ACE inhibitors, β-blockers, and diuretics are the most popular medications used to treat heart failure.

Table 6.8: Frequency of medication classes in the cohort

Medication class	**Frequency(N=5044)**
ACE inhibitor	55.5%
β blocker	48.6%
Angiotension Receptor Blocker	12.8%
Calcium Channel Blocker	4.1%
Statin use	43.2%
Diuretic use	68.7%
Allopurinol use	18.5%
Aldosterone Blocker	18.5%

6.2.3 Results and Discussions

This section presents the results of CPXR(Log) on HF risk prediction models, which are focused on four main aspects: (1) We compare the performance of CPXR(Log) against state- of-the-art classification algorithms such as Logistic Regression, Decision Tree, Support Vector Machine, Random Forest and AdaBoost. The results show that CPXR(Log) is much more accurate and outperforms other classifiers significantly. (2) We also present details on patterns and local models found by CPXR(Log) for HF risk prediction. Each pattern and the corresponding local model extracted by CPXR(Log) represent a distinct subgroup of patients with specific behaviors, whose survival risk should be calculated based on the local model assigned specifically to that subgroup of patients. Distinct pairs of patterns and local models are highly different from each other and they are highly different from the baseline model. (3) We show that the incorporation of co-morbidities extracted from EHR into our models improves the accuracy of CPXR(Log) and gives us more insights about the complexity of heart failure. We also show that the predictive power of co-morbidities has not been fully utilized by other classification algorithms in fact those algorithms produced less accurate prediction models when they use the co-morbidities as features for modeling building. (4) We examine the effect of the probabilistic loss function and compare it with the loss function used in [Taslimitehrani and Dong 2014].

CPXR(Log) depends on two parameters, minSup and ρ. In this study, we used fixed parameter values (minSup=0.03 and ρ=0.45). Regarding the other classifiers, we used their implementation in standard R packages [R core 2012]. Note that there are patient records with missing values for certain lab results and blood pressure measurements, a characteristic typical for real-world EHR data. We used multiple imputation to handle those missing values using a package called mi in R

Table 6.9: AUC of different classifiers

Algorithm	**1 year**	**2 years**	**5 years**
Decision Tree	0.66	0.5	0.5
Random Forest	0.8	0.72	0.72
Ada Boost	0.74	0.71	0.68
SVM	0.59	0.52	0.52
Logistic Regression	0.81	0.74	0.73
CPXR(Log)	**0.937**	**0.83**	**0.786**

[Su et al. 2011].

Table 6.10: Performance of different classifiers

Measure[4]	**Model**	**SVM**	**Log Reg.**	**CPXR(Log)**
Precision	1 year	0.66	0.74	**0.82**
	2 years	0.43	0.7	**0.78**
	5 years	0.2	0.51	**0.721**
Recall	1 year	0.7	0.65	**0.782**
	2 years	0.7	0.64	**0.76**
	5 years	0.5	0.5	**0.615**
Accuracy	1 year	0.75	0.88	**0.914**
	2 years	0.57	0.75	**0.83**
	5 years	0.66	0.71	**0.809**

As explained earlier our problem is classifying patients who survived after a diagnosis of HF vs those who did not survive using the CPXR(Log) algorithm based on EHR data. Hence, our outcome (response) variable is mortality status. In this study, we developed and validated three models to predict 1-, 2-, and 5- years survival in HF patients with the use of EHR extracted variables including demographic, vitals, lab results, medications, and co-morbidities.

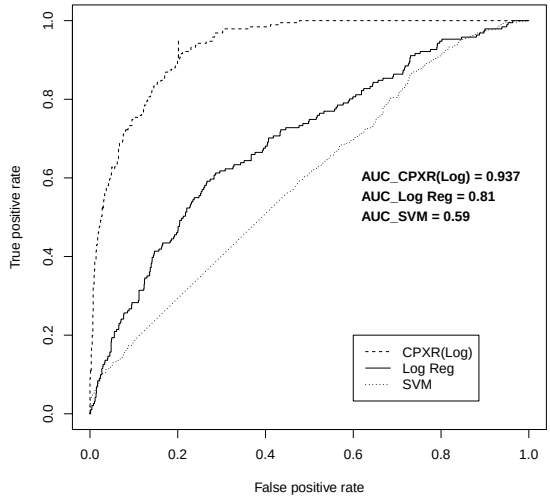
To enhance the generalizability of CPXR(Log) models, following common practice in clinical prediction modeling, we divide our dataset into two separate parts: a training part and a test part; the training dataset contains data for 1560 out of the 5044 patients, and the test dataset contains data for the remaining 3484 patients. The training and test datasets do not overlap.

We now compare the performance of CPXR(Log) against standard logistic regression and state-of-the-art classifiers concerning accuracy. Table 6.9 presents the AUC of the three models built by

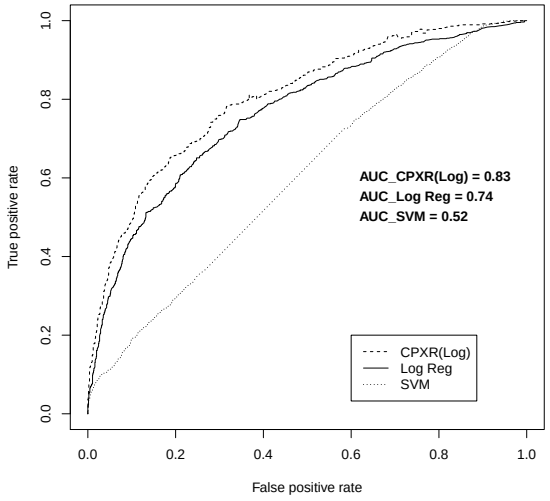
different classification algorithms. The results show that CPXR(Log) outperforms other classifiers consistently by large margins. The strong performance of CPXR(Log) implies there are highly diverse predictor-response relationships for HF patients and they were successfully extracted by CPXR(Log).

Furthermore, all CPXR(Log) models outperformed corresponding logistic regression and SVM models on all three other performance measures, as shown in Table 6.10. In Table 6.10, the cutoff values to optimize accuracy, precision and recall are determined in the way described in the previous section. In particular, CPXR(Log) improved the AUC of logistic regression, SVM, Random Forest and AdaBoost models by 15.6%, 58.8%, 17.1% and 26.6% on average, respectively. Further, Figures 6.4a, 6.4b and 6.4c show that the ROC curves of all three CPXR(Log) models have larger true positive rate for every false positive rate, than that of logistic regression and SVM models. Our results are also better than those reported by prior from Levy et al. [Levy et al. 2006]. Specifically, the AUC of 1- year model developed by CPXR(Log) is 5.6% larger than the most accurate model developed by Levy et al. [Levy et al. 2006].

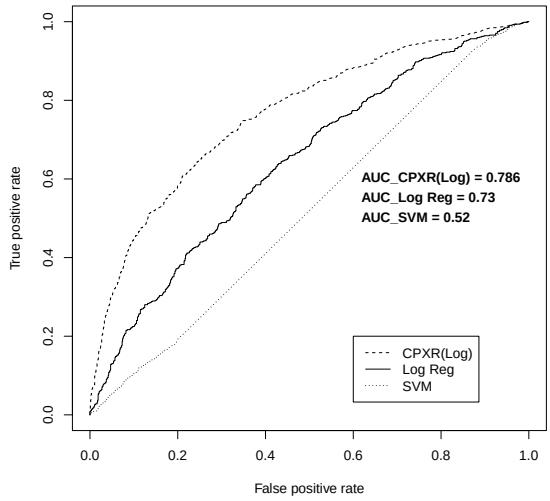
We now use our experimental results to discuss and highlight one of the main strengths of CPXR(Log), namely its ability to effectively utilize more predictor variables to derive more accurate models (a fact also observed in [Ghanbarian et al. 2015] when CPXR was used for linear regression), and to highlight the observation that large number of dimensions is also one of the challenges of EHR datasets (for traditional classification algorithms). EHR datasets often have a large number of variables and traditional classifiers fail to handle the high dimensionality data robustly, Table 6.11 demonstrates the extent to which AUC improved/decreased when more variables are used for CPXR(Log) and other classifiers. As we discussed earlier, we divided our predictor variables into four groups of Demographics and Vitals, labs, medications, and co-morbidities (Demographics and Vitals are in one group). We started with demographic and vital variables, and then in each step, added more variables in the model building process. In general, CPXR(Log) consistently produced better AUCs when more variables are added, and it also obtained larger improvement in most cases than other classifiers. Interestingly, other classifiers sometimes performed worse when additional variables were included in the models. For example, adding the 24 co-morbidity variables improved the AUC of CPXR(Log) models by 15.9%; in contrast, the accuracy of other classifiers did not improve in fact they decreased by 5.3% on average. It shows CPXR can effectively extract useful information capturing interactions among multiple predictor variables such as co-morbidities that are of-ten missed by other classification algorithms.



(a) ROC curve for one year models



(b) ROC curve for two years models



(c) ROC curve for five year models

Figure 6.4: ROC curves of one year, two years and five years models

6.2.4 Conclusion

We used CPXR(Log) to build heart failure survival prediction models, for 1-, 2-, and 5-years after HF is diagnosed based on EHR data. The models built by CPXR(Log) achieved much higher accuracy than standard logistic regression, Random Forest, SVM, decision tree and AdaBoost, which implies

Table 6.11: AUC improvement when more predictor variables are used by CPXR(Log) and other classifiers

Variable set change	**CPXR (Log)**	**Logistic Regression**	**Random Forest**	**SVM**	**Decision Tree**	**Boosting**
(Demo&Vital)→ (Demo&Vital)+Lab	4.8%	11.5%	19%	17.3%	0%	14.7%
(Demo&Vital)→ (Demo&Vital)+Lab+Med	8.9%	13.4%	21.2%	21.7%	0%	5.7%
(Demo&Vital)→ (Demo&Vital)+Lab+Med+Co-morbid	27.8%	9.6%	19.1%	19.5%	-10.4%	7.6%
(Demo&Vital)+Lab→ (Demo&Vital)+Lab+Med	3.2%	1.7%	1.7%	3.7%	0%	-9.8%
(Demo&Vital)+Lab→ (Demo&Vital)+Lab+Med+Co-morbid	20.9%	-1.7%	0%	1.8%	-10.4%	-8.1%
(Demo&Vital)+Lab+Med→ (Demo&Vital)+Lab+Med+Co-morbid	15.9%	-3.3%	-1.7%	-1.7%	-10.4%	1.8%

that there are fairly complicated interactions between predictor and response variables for heart failure. We also included 24 co-morbidities into our models and showed that adding these new variables gives us both more insights and improved accuracy of our models. In general, CPXR(Log) can effectively build highly accurate prediction models on datasets with diverse predictor-response relationships, but the other classification algorithms cannot effectively handle the high dimensionality and complexity of EHR data in order to build accurate prediction models. This study indicates that the behavior of HF patients is highly heterogeneous and that different patterns and local prediction models better suitable in predicting HF survival to properly handle the disease heterogeneity.

6.3 Application of CPXR in Saturated Hydraulic Conductivity

6.3.1 Introduction

Flow and transport modeling in subsurface flow and the hydrologic cycle requires fundamental characteristics of hydraulic properties. One of these hydraulic properties is soil water retention curve (SWRC) whose measurement, estimation, and even modeling continue to be under consideration in different communities, such as hydrology, soil science, and hydrogeology [Ghanbarian-Alavijeh and Millán 2010]. Sample dimensions, e.g., height (or length) and diameter influence soil water retention curve measurements. This (sample dimensions) effect on air-entry value might be one of the reasons that existing parametric pedotransfer functions estimate air-entry value from other easily available parameters inaccurately. In addition to SWRC, the saturated hydraulic conductivity (SHC) plays a key role in flow and solute transport modeling under saturated and unsaturated conditions. The effect of sample dimensions on the saturated hydraulic conductivity value attracted a great deal of attention. It has been shown that the SHC measurement is much influenced by sample size.

Since both soil water retention curve (SWRC) and saturated hydraulic conductivity (SHC) measurements are time consuming, indirect methods have been developed to estimate SWRC and SHC from other available properties, such as sand, silt, and clay contents, organic matter, bulk density, and particle-size distribution. Observations indicate the need to pay more attention to the effects of scale on soil hydraulic properties and the need to include such effects in pedotransfer functions. However, to our knowledge up to now no research has focused on the effect of sample dimensions upon the development of the soil hydraulic properties pedotransfer functions. Therefore, the main objective of this study was to investigate the sample dimensions effect on the prediction of the soil water retention curve (SWRC) and the saturated hydraulic conductivity (SHC). To achieve this goal we evaluate a new data mining approach called contrast pattern aided regression (CPXR) that was applied to soil hydraulic properties predictions using data available in the UNSODA database.

6.3.2 Datasets

In this study, we select disturbed and (mostly) undisturbed laboratory experiments from the UNSODA database whose sample dimensions e.g., height (or length) and internal diameter were available and two datasets including 210 and 213 laboratory samples, were formed to develop and evaluate pedotransfer functions for the soil water retention curve (SWRC) and the saturated hydraulic conductivity (SHC), respectively.

The van Genuchten soil water retention curve model [van Genuchten 1980] parameters, such as α, n, θ_r, and θ_s reported in the ROSETTA database [Schaap et al. 2001] were used to calculate water contents at different tension heads e.g., inflection point , 10, 33, 50, 100, 300, 500, 1000, 1500 kPa for each soil sample.

To develop both point and parametric pedotransfer functions for SWRC, 6 input variables, such as sand, silt, and clay contents, bulk density, geometric mean diameter d_g, and geometric standard deviation δ_g of particles were used (hereafter SWRC1). The last two parameters (i.e., d_g and δ_g in mm) were determined from clay, silt, and sand contents [Shirazi and Boersma 1984]. In order to investigate the effect of sample dimensions on the development and evaluation of the pedotransfer functions, in addition to those 6 variables, sample internal diameter (ID) and height or length (L) were also included as input variables (hereafter SWRC2). For a few samples with non-circular cross sectional area, the equivalent diameter was determined. In addition to point pedotransfer functions, we also developed parametric pedotransfer functions to predict the van Genuchten soil water retention curve model parameters, such as θ_r, θ_s, α, and n from available soil properties. The same input variables introduced to SWRC1 and SWRC2 models were used to develop two parametric models (hereafter SWRC3 and SWRC4). The difference between the SWRC3 and SWRC4 models is that two more input variables e.g., sample internal diameter (ID) and height or length (L) were used to develop the SWRC4 model.

For the development of the SHC pedotransfer functions, four models were considered: SHC1 included input variables, such as sand, silt, and clay contents, geometric mean diameter, geometric standard deviation, and bulk density. SHC2 consisted of two extra input variables e.g., sample height or length and internal diameter. In SHC3, besides textural data e.g., sand, silt, and clay contents, geometric mean diameter, geometric standard deviation, and bulk density, vG soil water retention curve model parameters e.g., α, n, θ_r, and θ_s as well as effective porosity were also used. SHC4 included sample diameter and height or length in addition to all input variables applied in SHC3. To develop pedotransfer functions for saturated hydraulic conductivity, the log-transformed SHC values were used in the training and testing processes. Table 6.12 summarizes the input and output variables for all models developed in this study.

6.3.3 Results and Discussion

6.3.3.1 Point Pedotransfer Functions for Soil Water Retention Curve (SWRC)

The results obtained for the CPXR method indicate that including sample internal diameter (ID) and height or length (L) as input variables increased considerably the accuracy and reliability of the developed pedotransfer functions. For example, at θ_s and θ_i the RMSE values in the testing

Table 6.12: Input and output variables of different models developed in this study.

Model	**Input variables**[5]	**Output variables**
SWRC1[6](point)	Sa, Si, Cl, ρ_b, d_g, σ_g	θ_s, θ_i, θ_{10}, θ_{30}, θ_{50}, θ_{100}, θ_{300}, θ_{500}, θ_{1000}, θ_{1500}
SWRC2(point)	Sa, Si, Cl, ρ_b, d_g, σ_g, ID, L	θ_s, θ_i, θ_{10}, θ_{30}, θ_{50}, θ_{100}, θ_{300}, θ_{500}, θ_{1000}, θ_{1500}
SWRC3 (param)	Sa, Si, Cl, ρ_b, d_g, σ_g	θ_r, θ_s, $log_e(\alpha)$, $log_e(n)$
SWRC4 (param)	Sa, Si, Cl, ρ_b, d_g, σ_g, ID, L	θ_r, θ_s, $log_e(\alpha)$, $log_e(n)$
SHC1[7]	Sa, Si, Cl, ρ_b, d_g, σ_g	$log_e(K_{sat})$
SHC2	Sa, Si, Cl, ρ_b, d_g, σ_g, ID, L	$log_e(K_{sat})$
SHC3	Sa, Si, Cl, ρ_b, d_g, σ_g, θ_r, θ_s, α, n, ϕ_e	$log_e(K_{sat})$
SHC4	Sa, Si, Cl, ρ_b, d_g, σ_g, θ_r, θ_s, α, n, ϕ_e, ID, L	$log_e(K_{sat})$

(training) process decreased only by 11% (15%) and 15% (7%), respectively, while at θ_{10} and θ_{1500} the RMSE values decreased by 40% (31%) and 23% (32%) after we included sample dimensions (i.e., ID and L).

Comparing the R^2 of the SWRC1 and SWRC2 models (presented in Table 6.13) for the training and testing processes also show that the accuracy of the pedotransfer functions increased with including two sample internal diameter and height or length variables.

The graphical results of the SWRC1 and SWRC2 models developed using the CPXR method (the predicted water content as a function of the measured one at different tension heads) for one iteration subset (or one fold) are shown in Figure 6.5. The RMSE values presented in Figure 6.5 are not comparable since the obtained results are not from the same split. However, as was demonstrated in Table 6.13, sample internal diameter (ID) and height or length (L) play a nontrivial role in the prediction of the soil water content at different tension heads.

The obtained results of the multiple linear regression (MLR) method used in this study for models SWRC1 and SWRC2 are presented in Table 6.14. In the most improved case (see θ_{1000} in Table 6.14), including ID and L as input variables could reduce (increase) the RMSE (R^2) value only by 2.4% (1.0%). Comparison of the RMSE and R^2 values of SWRC1 with those of SWRC2 implies that including sample dimensions, e.g., sample diameter and height or length did not improve the accuracy and reliability of the pedotransfer functions. This means the MLR approach, in contrast to the CPXR technique, is not capable to detect interactions between input variables that define different subgroups of data with highly distinct predictor-response relationships, and to extract nonlinear patterns among input and output variables (see Table 6.13). In support, the RMSE values given in Table 6.13 (resulted from CPXR) are 45-74% less than those reported in Table 6.14 (resulted from MLR) for both SWRC1 and SWRC2 models and the training and testing processes.

Comparison of the predicted water content with the measured one using the CPXR and MLR

Table 6.13: Statistical parameters ($RMSE$ and R^2) calculated for the training and testing splits and point pedotransfer functions of soil water retention curve (SWRC1 and SWRC2) using the CPXR method.

	Model	θ_s	θ_i	θ_{10}	θ_{30}	θ_{50}	θ_{100}	θ_{300}	θ_{500}	θ_{1000}	θ_{1500}
						RMSE					
Training	SWRC1	0.018	0.013	0.030	0.025	0.023	0.020	0.017	0.017	0.019	0.021
	SWRC2	0.016	0.011	0.018	0.018	0.018	0.017	0.014	0.013	0.014	0.016
Testing	SWRC1	0.019	0.013	0.029	0.026	0.023	0.021	0.018	0.019	0.020	0.025
	SWRC2	0.016	0.012	0.020	0.019	0.019	0.022	0.020	0.016	0.015	0.017
						R^2					
Training	SWRC1	0.96	0.98	0.96	0.96	0.97	0.97	0.97	0.97	0.96	0.95
	SWRC2	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.97
Testing	SWRC1	0.94	0.97	0.97	0.94	0.97	0.97	0.95	0.96	0.95	0.94
	SWRC2	0.95	0.96	0.94	0.95	0.97	0.96	0.95	0.98	0.97	0.94

methods for SWRC1 and SWRC2 models indicates that water content values predicted by MLR are more scattered than those predicted by CPXR, demonstrating the higher reliability of the proposed method (CPXR) in this study (results not shown). Comparison of the results presented in Tables 6.13 and 6.14 shows that indeed different relationships control the sample dimensions effects in different parts of the databases that can be captured by different patterns and the CPXR approach.

6.3.3.2 Parametric Pedotransfer Functions for Soil Water Retention Curve (SWRC):

The RMSE, and R^2 values for the two SWRC3 and SWRC4 models developed using the CPXR method are given in Table 6.15. Comparison of the RMSE and RMSLE values of the SWRC3 and SWRC4 models indicates that including sample dimensions increased the accuracy and reliability of the developed pedotransfer functions in prediction of θ_s, θ_r, $loge(\alpha)$ and $loge(n)$, for both training and testing processes. Particularly, in the training (testing) process, the RMSE and RMSLE values of the pedotransfer functions developed for θ_r, $loge(\alpha)$, and $loge(n)$ parameters decreased by 20% (21%), 35% (35%), and 49% (50%), respectively (see Table 6.15). However, the RMSE value of those developed for θ_s reduced only by 11% (16%). The obtained results confirm the nontrivial effect of the sample dimensions on the soil water retention curve prediction, since both α and n parameters describe the shape of the soil water retention curve.

Table 6.15 also presents the R^2 values for the two SWRC3 and SWRC4 pedotransfer functions developed by the CPXR method. Comparison of the SWRC3 and SWRC4 models demonstrate

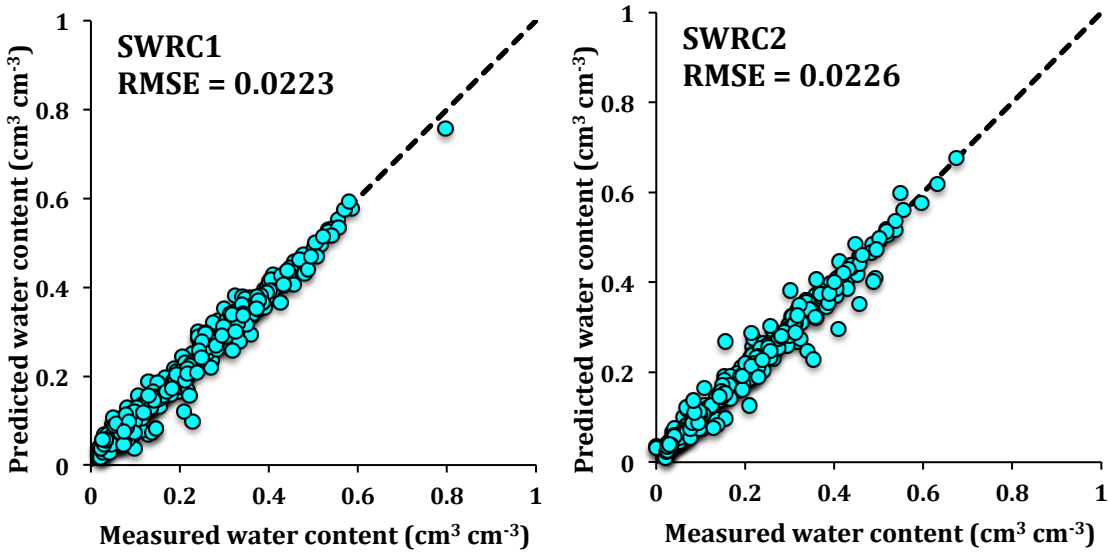


Figure 6.5: Predicted water content versus measured one using the CPXR method for the SWRC1 and SWRC2 point pedotransfer functions developed in this study

that the R^2 value of the pedotransfer function developed for $loge(n)$ increased considerably from 0.86 to 0.96 (12% increase) and 0.81 to 0.93 (15% increase) in the training and testing processes, respectively. This means that the accurate prediction of the shape of the soil water retention curve requires information on the sample dimensions.

The MLR results of the training and testing processes are also given in Table 6.15. We found that the accuracy (RMSE) of the pedotransfer functions developed using the MLR method only improved by 5% and 4% in the training and testing processes, respectively, when sample dimensions were included. Our results imply that the MLR method failed to detect patterns and construct nonlinear connections between input and output variables properly. However, the CPXR approach found effectively the nonlinear structures between inputs and outputs (see Table 6.15).

6.3.4 Conclusion

In this study, pedotransfer functions were developed using contrast pattern aided regression (CPXR) and multiple linear regression (MLR) methods to predict soil water retention curve and saturated hydraulic conductivity. For this purpose, the 10-fold cross-validation approach was applied to evaluate the developed models accuracy and reliability. The obtained results indicated that the CPXR method predicted output variables more accurately than the MLR technique in both train and test steps. As expected, we demonstrated that inclusion of sample internal diameter and length could

Table 6.14: Statistical parameters ($RMSE$ and R^2) calculated for the training and testing splits and point pedotransfer functions of soil water retention curve (SWRC1 and SWRC2) using the MLR method.

	Model	θ_s	θ_i	θ_{10}	θ_{30}	θ_{50}	θ_{100}	θ_{300}	θ_{500}	θ_{1000}	θ_{1500}
						RMSE					
Training	SWRC1	0.062	0.042	0.062	0.048	0.045	0.041	0.039	0.039	0.041	0.045
	SWRC2	0.061	0.042	0.061	0.048	0.045	0.041	0.039	0.039	0.040	0.045
Testing	SWRC1	0.062	0.042	0.062	0.048	0.045	0.041	0.039	0.040	0.041	0.045
	SWRC2	0.061	0.042	0.062	0.049	0.045	0.041	0.040	0.039	0.041	0.045
						R^2					
Training	SWRC1	0.81	0.76	0.81	0.87	0.88	0.88	0.87	0.86	0.84	0.80
	SWRC2	0.82	0.76	0.81	0.87	0.88	0.88	0.87	0.86	0.84	0.80
Testing	SWRC1	0.79	0.73	0.77	0.84	0.85	0.84	0.83	0.84	0.81	0.77
	SWRC2	0.79	0.72	0.77	0.85	0.84	0.84	0.84	0.83	0.80	0.78

Table 6.15: Statistical parameters calculated for train and test splits and parametric pedotransfer functions of the van Genuchten soil water retention curve model (SWRC3 and SWRC4) using the CPXR and MLR approaches.

			θ_r	θ_s	$log_e(\alpha)$	$log_e(n)$	θ_r	θ_s	$log_e(\alpha)$	$log_e(n)$
Method		Model	RMSE		RMSELE		R2			
CPXR	Train	SWRC3	0.030	0.018	0.531	0.191	0.80	0.96	0.83	0.86
		SWRC4	0.024	0.016	0.346	0.098	0.87	0.97	0.93	0.96
	Test	SWRC3	0.034	0.019	0.570	0.201	0.76	0.94	0.83	0.81
		SWRC4	0.027	0.016	0.371	0.101	0.83	0.95	0.90	0.93
MLR	Train	SWRC3	0.060	0.055	1.136	0.328	0.19	0.60	0.21	0.69
		SWRC4	0.060	0.054	1.158	0.319	0.19	0.62	0.18	0.61
	Test	SWRC3	0.061	0.055	1.140	0.330	0.21	0.58	0.20	0.58
		SWRC4	0.060	0.053	1.167	0.324	0.22	0.59	0.18	0.59

improve the accuracy and reliability of the developed pedotransfer functions remarkably.

7

Conclusion

In this chapter, we summarize the findings of this dissertation.

7.1 Summary

Regression and classification techniques play an important role in our daily life. Identifying patients at high risk of developing a disease, evaluating trends and sales estimates in business word, forecasting customer demand to drive holistic execution, identifying high-risk drivers in insurances business are some of the applications of regression and classification. Accuracy and interpretability are two important and contradictory goals in designing regression and classification techniques.

Regression and classification are an ongoing and challenging field of research, specifically when it comes to heterogeneous and complex datasets. So far, there are an enormous number of regression and classification techniques, but most of them often fail to produce highly accurate and interpretable models. In this dissertation, we deeply investigated the design issues of traditional regression and classification techniques and proposed a series of solutions to meet those challenges.

We articulated the concept of predictor-response (PR) relationship, which represents multi-dimensional interactions between predictor and response variables. The Majority of the traditional regression and classification techniques are unable to extract and model those high dimensional interactions. Our proposed methodology to extract PR relationships are based on two key ideas: First, we used the concept of patterns to logically characterize the subgroups of data points. Then we use the local regression or classification models to behaviorally characterize the hidden predictor-response relationships in those subgroups of data points. Second, we incorporate a small set of patterns and local model pairs to form a pattern aided regression (PXR) for numerical prediction problems (regression) and pattern aided classification(PXC) for classification problems. Each pair of pattern and local model represents a specific PR relationship, and a set of pattern and local model

pairs reveals a set of diverse PR relationships in a heterogeneous dataset.

In chapters 4 and 5, we proposed two algorithms called contrast pattern aided regression (CPXR) and contrast pattern aided classification (CPXC) to build PXR and PXC models, respectively. In both CPXR and CPXC, we performed a systematic evaluation to measure the performance of these techniques. Experimental results revealed that both techniques outperformed state-of-the-art regression and classification techniques often by big margins. We also investigated overfitting phenomena and sensitivity to noise. Our experimental results indicated both CPXR and CPXC does not overfit in comparison to other algorithms.

We also studied the problem of classification on imbalanced datasets. In chapter 5, we adopted CPXC methodology to handle classification on imbalanced datasets and proposed a new classifier called CPXC on Imbalanced Datasets (CPXCim). In CPXCim, we introduced a new filtering method to identify imbalanced local models and used a weighting method to promote minority class instances. Experimental results performed on a set of benchmark datasets present very good performance comparing to other imbalanced classifiers.

In chapter 6, we applied CPXR and CPXC on several applications. In the first application, we applied CPXC on a trial dataset to predict patient's outcome within 6 months after traumatic brain injury (TBI). CPXC improved the AUC of logistic regression by 11.7%. In the second application, CPXC is applied on an EHR dataset to measure the risk of heart failure on 1- 2, and 5- years. CPXC outperformed other classifiers such as SVM and decision tree. In the last application, we used CPXR to predict soil water retention curve and saturated hydraulic conductivity.

Overall, CPXR, CPXC, and CPXC-im showed huge advancements in accuracy and interpretability. As a result, the outcome of classification or regression will be more reliable using these methods.

Appendices

Table 1: Symbols table

Symbol	**Meaning**	**Symbol**	**Meaning**
D	Dataset	$x_1, ...x_k$	A set of predictor variables
y	response variable	X_i	A vector of predictor variables
CPXR	Contrast Pattern Aided Regression	CPXC	Contrast Pattern Aided Classification
PXR	Pattern Aided Regression	PXR	Pattern Aided Classification
RMSE	Root Mean Square error	PLR	Piecewise Linear Regression
SVR	Support Vector Regression	SVM	Support Vector Machine
BART	Bayesian Additive Regression Trees	GBM	Gradient Boosting Method
NBC	Naive Bayesian Classifier	DT	Decision Tree
SLogR	Standard Logistic Regression	ϵ	Regression random error
β	Regression coefficient	r	Residual value
mds	matching dataset	$Supp$	Support
$minSup$	Minimum support	$Supp_{ratio}$	Support ratio
f_i	Local regression model	f_d	Default regression model
h_i	Local classifier	h_d	Default classifier
P	Pattern	w_i	Local model's weight
PR	Predictor-Response Relationship	k	Number of patterns in PS
PS	Pattern set	CPS	Contrast Pattern Set
PIP	Positive improvement patterns	$r_x(f)$	f's residual on an instance x
TER	Total Error Reduction	TRR	Total Residual Reduction
AER	Average Error Reduction	ARR	Average Residual Reduction
ρ	Splitting point	κ	Index of splitting instance
γ	Support ratio threshold		
EC	Equivalence Class	MG	Minimal Generator
LE	Large Error instances	SE	Small Error instances

References

HHS Initiative on Multiple Chronic Conditions. http://www.hhs.gov/ash/initiatives/mcc/.

Adam J. Grove, D. S. 1998. Boosting in the limit: Maximizing the margin of learned ensembles. *AAAI/IAAI*, 692–699.

Agrawal, R., Mannila, H., and Srikant, R. 1996. Fast Discovery of Association Rules. *Advances in knowledge discovery and data mining 12,* 1, 307–328.

Akbilgic, O., Bozdogan, H., and Balaban, M. E. 2013. A novel Hybrid RBF Neural Networks model as a forecaster. *Statistics and Computing 24,* 3 (feb), 365–375.

Antonie, M. and Zaiane, O. 2002. Text document categorization by term association. *IEEE International Conference on Data Mining, ICDM 2003*, 19–26.

Auer, J. and Bajorath, J. 2006. Emerging chemical patterns: a new methodology for molecular classification and compound selection. *Journal of chemical information and modeling 46,* 6 (jan), 2502–14.

Bache, K. and Lichman, M. 2013. UCI machine learning repository.

Bagley, S. C., White, H., and Golomb, B. A. 2001. Logistic regression in the medical literature:. *Journal of Clinical Epidemiology 54,* 10 (oct), 979–985.

Batista, G., Prati, R., and Monard, M. 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter 6,* 1, 20–29.

Breiman, L. 1996. Bagging predictors. *Machine Learning 24,* 2 (aug), 123–140.

Breiman, L. 2001. Random Forests. *Machine Learning 45,* 1, 5–32.

Breiman, L., Friedman, J., and Charles, S. J. 1984. Classification and Regression Trees. *CRC press*.

Bringmann, B. and Zimmermann, A. 2008. One in a million: picking the right patterns. *Knowledge and Information Systems 18,* 1 (mar), 61–81.

Brown, A. W., Malec, J. F., McClelland, R. L., Diehl, N. N., Englander, J., and Cifu, D. X. 2005. Clinical elements that predict outcome after traumatic brain injury: a prospective multicenter recursive partitioning (decision-tree) analysis. *Journal of neurotrauma 22,* 10 (oct), 1040–51.

CDC. Centers for Disease Control and Prevention. http://www.cdc.gov/.

Chawla, N. and Bowyer, K. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 321–357.

Chen, C., Liaw, A., and Breiman, L. 2004. Using random forest to learn imbalanced data. *University of California, Berkeley*.

Chen, L. and Dong, G. 2006. Masquerader Detection Using OCLEP: One-Class Classification Using Length Statistics of Emerging Patterns. In *2006 Seventh International Conference on Web-Age Information Management Workshops*. IEEE, 5–5.

Chipman, H. and McCulloch, R. 2009. BayesTree: Bayesian Methods for Tree Based Models. *URL http://CRAN. R-project. org/package= BayesTree. R package version 0.3-1*.

Chipman, H. a., George, E. I., and McCulloch, R. E. 2012. BART: Bayesian additive regression trees. *Annals of Applied Statistics 6,* 1, 266–298.

Cohen, G., Hilario, M., and Sax, H. 2006. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine 37,* 1, 7–18.

CORNFIELD, J. 1951. A method of estimating comparative rates from clinical data; applications to cancer of the lung, breast, and cervix. *Journal of the National Cancer Institute 11,* 6 (jun), 1269–75.

CRASH Trial Collaborators. 2005. Final results of MRC CRASH, a randomised placebo-controlled trial of intravenous corticosteroid in adults with head injuryoutcomes at 6 months. *The Lancet 365,* 9475, 1957–1959.

Dekel, O. and Shamir, O. 2012. There's a Hole in My Data Space: Piecewise Predictors for Heterogeneous Learning Problems. In *International Conference on Artificial Intelligence and Statistics*. 291–298.

Domingos, P. 1999. Metacost: A general method for making classifiers cost-sensitive. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 155–164.

Dong, G. and Bailey, J. 2012. *Contrast Data Mining: Concepts, Algorithms, and Applications.* CRC Press.

Dong, G. and Li, J. 1999. Efficient mining of emerging patterns. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '99.* ACM Press, New York, New York, USA, 43–52.

Dong, G. and Taslimitehrani, V. 2015. Pattern-Aided Regression Modeling and Prediction Model Analysis. *IEEE Transactions on Knowledge and Data Engineering 27,* 9, 2452–2465.

Dong, G. and Taslimitehrani, V. 2016. Pattern Aided Classification. In *2016 SIAM Internatioanl Conference on Data Mining.*

Dong, G., Zhang, X., Wong, L., and Li, J. 1999. CAEP: Classification by aggregating emerging patterns. In *Discovery Science.* Springer, 30–42.

Fayyad, U. M. and Irani, K. B. Multi-interval discretization of continuous-valued attribute for classification learning.

Fenghua, W., Jihong, X., Zhifang, H., and Xu, G. 2014. Stock Price Prediction based on SSA and SVM. *Procedia Computer Science 31*, 625–631.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. 2014. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research 15,* 1 (jan), 3133–3181.

Freund, Y. and Schapire, R. E. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences 55,* 1 (aug), 119–139.

Friedman, J., Hastie, T., and Tibshirani, R. 2009. glmnet: Lasso and elastic-net regularized generalized linear models. *R package version.*

Friedman, J. H. 1991. Multivariate Adaptive Regression Splines. *The Annals of Statistics 19,* 1 (mar), 1–67.

Friedman, J. H. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis 38,* 4 (feb), 367–378.

Galton, F. 1886. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland 15*, 246–263.

Ghanbarian, B., Taslimitehrani, V., Dong, G., and Pachepsky, Y. A. 2015. Sample dimensions effect on prediction of soil water retention curve and saturated hydraulic conductivity. *Journal of Hydrology 528*, 127–137.

Ghanbarian-Alavijeh, B. and Millán, H. 2010. Point pedotransfer functions for estimating soil water retention curve. *International Agrophysics 24,* 3, 243–251.

Gu, Q. and Han, J. 2013. Clustered Support Vector Machines. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*. 307–315.

Han, J., Kamber, M., and Pei, J. 2011. *Data Mining: Concepts and Techniques*. Elsevier.

Hastie, T., Tibshirani, R., and Friedman, J. H. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

Hawkins, D. M. 2004. The problem of overfitting. *Journal of chemical information and computer sciences 44,* 1, 1–12.

Hosmer, D. W., Jr., Lemeshow, S., and Sturdivant, R. X. 2013. *Applied Logistic Regression*. John Wiley & Sons.

Hosmer, D. W. and Lemeshow, S. 2000. *Applied Logistic Regression*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Hudson, D. J. 1996. Fitting Segmented Curves Whose Join Points Have to Be Estimated. *Journal of the American Statistical Association 61,* 316 (apr), 1097–1129.

Hukkelhoven, C. W. P. M., Steyerberg, E. W., Habbema, J. D. F., Farace, E., Marmarou, A., Murray, G. D., Marshall, L. F., and Maas, A. I. R. 2005. Predicting outcome after traumatic brain injury: development and validation of a prognostic score based on admission characteristics. *Journal of neurotrauma 22,* 10 (oct), 1025–39.

Hukkelhoven, C. W. P. M., Steyerberg, E. W., Rampen, A. J. J., Farace, E., Habbema, J. D. F., Marshall, L. F., Murray, G. D., and Maas, A. I. R. 2003. Patient age and outcome following severe traumatic brain injury: an analysis of 5600 patients. *Journal of neurosurgery 99,* 4 (oct), 666–73.

IMPACT. IMPACT: International Mission for Prognosis and Analysis of Clinical Trials in TBI.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. 1991. Adaptive Mixtures of Local Experts. *Neural Computation 3,* 1 (feb), 79–87.

Jennett, B., Teasdale, G., Braakman, R., Minderhoud, J., and Knill-Jones, R. 1976. Predicting outcome in individual patients after severe head injury. *Lancet 1,* 7968 (may), 1031–4.

Johansson, U., Sönströd, C., Norinder, U., and Boström, H. 2011. Trade-off between accuracy and interpretability for predictive in silico modeling. *Future medicinal chemistry 3,* 6 (apr), 647–63.

Kearns, M. and Valiant, L. 1994. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM 41,* 1 (jan), 67–95.

Kearns, M. J. and Mansour, Y. 1998. A Fast, Bottom-Up Decision Tree Pruning Algorithm with Near-Optimal Generalization. In *ICML*. Vol. 98. Citeseer, 269–277.

Kennedy, R. E., Livingston, L., Riddick, A., Marwitz, J. H., Kreutzer, J. S., and Zasler, N. D. Evaluation of the Neurobehavioral Functioning Inventory as a depression screening tool after traumatic brain injury. *The Journal of head trauma rehabilitation 20,* 6 (jan), 512–26.

Kim, H., Loh, W.-Y., Shih, Y.-S., and Chaudhuri, P. 2007. Visualizable and interpretable regression models with good prediction power. *IIE Transactions 39,* 6 (mar), 565–579.

Kubat, M. and Matwin, S. 1997. Addressing the curse of imbalanced data sets: One sided sampling. *Proc. of the Int'l Conf. on Machine Learning*.

Kuncheva, L. I. 2004. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons.

Letham, B., Rudin, C., McCormick, T. H., and Madigan, D. 2013. An Interpretable Stroke Prediction Model using Rules and Bayesian Analysis.

Levy, W. C., Mozaffarian, D., Linker, D. T., Sutradhar, S. C., Anker, S. D., Cropp, A. B., Anand, I., Maggioni, A., Burton, P., Sullivan, M. D., Pitt, B., Poole-Wilson, P. A., Mann, D. L., and Packer, M. 2006. The Seattle Heart Failure Model: prediction of survival in heart failure. *Circulation 113,* 11 (mar), 1424–33.

Li, D., Liu, C., and Hu, S. 2010. A learning method for the class imbalance problem with medical data sets. *Computers in biology and medicine 40,* 5, 509–518.

Li, H., Li, J., Wong, L., Feng, M., and Tan, Y.-P. 2005. Relative risk and odds ratio: A Data Mining Perspective (Corrected Version). ACM Press, New York, New York, USA.

Li, J., Dong, G., Ramamohanarao, K., and Wong, L. 2004. Deeps: A new instance-based lazy discovery and classification system. *Machine Learning*.

Li, J., Liu, G., and Wong, L. 2007. Mining statistically important equivalence classes and delta-discriminative emerging patterns. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*. ACM Press, New York, New York, USA, 430.

Li, J. and Wong, L. 2002. Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. *Bioinformatics 18,* 5 (may), 725–734.

Li, W., Han, J., and Pei, J. 2001. CMAR: accurate and efficient classification based on multiple class-association rules. In *Proceedings 2001 IEEE International Conference on Data Mining*. IEEE Comput. Soc, 369–376.

Liaw, A. and Wiener, M. 2002. Classification and Regression by randomForest. *R News 2,* 3, 18–22.

Ling, C., Yang, Q., Wang, J., and Zhang, S. 2004. Decision trees with minimal costs. *Proceedings of the twenty-first international conference on Machine learning*, 69.

Ling, C. X. and Li, C. 1998. Data Mining for Direct Marketing: Problems and Solutions. *KDD 98*.

Liu, Q. and Dong, G. 2009. A Contrast Pattern Based Clustering Quality Index for Categorical Data. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 860–865.

Loyola-González, O. and Al., E. 2016. Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases. *Neurocomputing 175*, 935–947.

Luellen, J. K., Shadish, W. R., and Clark, M. H. 2005. Propensity scores: an introduction and experimental test. *Evaluation review 29,* 6 (dec), 530–58.

Maas, A. I. R., Marmarou, A., Murray, G. D., Teasdale, S. G. M., and Steyerberg, E. W. 2007. Prognosis and clinical trial design in traumatic brain injury: the IMPACT study. *Journal of neurotrauma 24,* 2 (feb), 232–8.

Mao, S. and Dong, G. 2005. Discover of Highly Differentiative Gene Groups from Microarray Gene Expression Data Using The Gene Club Approach. *Journal of Bioinformatics and Computational Biology 03,* 06 (dec), 1263–1280.

McCullagh, P. 1984. Generalized linear models. *European Journal of Operational Research 37*.

McZgee, V. E. and Carleton, W. T. 1970. Piecewise Regression. *Journal of the American Statistical Association 65,* 331 (sep), 1109–1124.

Meir, R. and Ratsch, G. 2002. An introduction to boosting and leveraging. *Knowledge Discovery In Databases: Pkdd 2005 2600*, 118–183.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. 2012. e1071: Misc Functions of the Department of Statistics (e1071), TU Wien, 2012. *R package version*, 1–6.

Montgomery, D. C., Peck, E. A., and Vining, G. G. 2015. *Introduction to Linear Regression Analysis*. Wiley.

Muggeo, V. M. R. 2008. Segmented: an R package to fit regression models with broken-line relationships. *R news 8,* 1, 20–25.

Murray, G. D., Butcher, I., McHugh, G. S., Lu, J., Mushkudiani, N. A., Maas, A. I. R., Marmarou, A., and Steyerberg, E. W. 2007. Multivariable prognostic analysis in traumatic brain injury: results from the IMPACT study. *Journal of neurotrauma 24,* 2 (feb), 329–37.

N, A. K. and Sheshadri, H. 2012. On the classification of imbalanced datasets. *International Journal of Computer Applications 44*.

Natekin, A. and Knoll, A. 2013. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics 7*, 21.

National Center for Biotechnology and Medicine, U. S. N. L. o. M. 2011. Pubmed - NCBI. *2011,* 2. February, http://www.ncbi.nlm.nih.gov/pubmed.

Orriols-Puig, A. and Bernadó-Mansilla, E. 2009. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing 13,* 3, 213–225.

Ouwerkerk, W., Voors, A. A., and Zwinderman, A. H. 2014. Factors influencing the predictive power of models for predicting mortality and/or heart failure hospitalization in patients with heart failure. *JACC. Heart failure 2,* 5 (oct), 429–36.

Panahiazar, M., Taslimitehrani, V., Pereira, N. L., and Pathak, J. 2015a. Using EHRs and Machine Learning for Heart Failure Survival Analysis. *Proceedings of the 15th World Congress on Health and Biomedical Informatics: MEDINFO 2015: EHealth-enabled Health 216*, 40.

Panahiazar, M., Taslimitehrani, V., Pereira, N. L., and Pathak, J. 2015b. Using EHRs for Heart Failure Therapy Recommendation Using Multidimensional Patient Similarity Analytics. *Proceedings of MIE2015: Digital Healthcare Empowering Europeans 210*, 369.

Pazzani, M., Merz, C., and Murphy, P. 1994. Reducing misclassification costs. *Proceedings of the Eleventh International Conference on Machine Learning*, 217–225.

Perel, P., Arango, M., Clayton, T., Edwards, P., Komolafe, E., Poccock, S., Roberts, I., Shakur, H., Steyerberg, E., and Yutthakasemsunt, S. 2008. Predicting outcome after traumatic brain injury: practical prognostic models based on large cohort of international patients. *BMJ (Clinical research ed.) 336,* 7641 (feb), 425–9.

Perel, P., Edwards, P., Wentz, R., and Roberts, I. 2006. Systematic review of prognostic models in traumatic brain injury. *BMC medical informatics and decision making 6,* 1 (jan), 38.

Perel, P., Wasserberg, J., Ravi, R. R., Shakur, H., Edwards, P., and Roberts, I. 2007. Prognosis following head injury: a survey of doctors from developing and developed countries. *Journal of evaluation in clinical practice 13,* 3 (jun), 464–5.

Quinlan, J. 1986. Induction of decision trees. *Machine learning 1,* 1, 81–106.

Quinlan, J. R. 1993. C4.5: programs for machine learning.

R core, T. 2012. A Language and Environment for Statistical Computing. R Foundation for Statistical Computing.

Ridgeway, G. 2006. gbm: Generalized boosted regression models. *R package version 1,* 3.

Schaap, M. G., Leij, F. J., and van Genuchten, M. T. 2001. rosetta: a computer program for estimating soil hydraulic parameters with hierarchical pedotransfer functions. *Journal of Hydrology 251,* 3-4 (oct), 163–176.

Schapire, R. E. 1990. The strength of weak learnability. *Machine Learning 5,* 2 (jun), 197–227.

Seber, G. A. F. and Wild, C. J. 2003. *Nonlinear Regression.* John Wiley & Sons.

Segal, M. R. 2004. Machine Learning Benchmarks and Random Forest Regression. *Center for Bioinformatics & Molecular Biostatistics*.

Shirazi, M. A. and Boersma, L. 1984. A Unifying Quantitative Analysis of Soil Texture. *Soil Science Society of America Journal 48,* 1 (aug), 142.

Smola, A. and Vapnik, V. 1997. Support vector regression machines. *Advances in Neural Information Processing Systems* 9, 155–161.

Specht, D. F. 1991. A general regression neural network. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council 2,* 6 (jan), 568–76.

Steyerberg, E. 2008. *Clinical Prediction Models: A Practical Approach to Development, Validation, and Updating.* Springer Science & Business Media.

Steyerberg, E. W., Mushkudiani, N., Perel, P., Butcher, I., Lu, J., McHugh, G. S., Murray, G. D., Marmarou, A., Roberts, I., Habbema, J. D. F., and Maas, A. I. R. 2008. Predicting outcome after traumatic brain injury: development and international validation of prognostic scores based on admission characteristics. *PLoS medicine 5,* 8 (aug), e165.

Su, Y.-S., Yajima, M., Gelman, A. E., and Hill, J. 2011. Multiple Imputation with Diagnostics (mi) in R: Opening Windows into the Black Box. *Journal of Statistical Software 45,* 2 (dec), 1–31.

Taslimitehrani, V. and Dong, G. 2014. A New CPXR Based Logistic Regression Method and Clinical Prognostic Modeling Results Using the Method on Traumatic Brain Injury. In *2014 IEEE International Conference on Bioinformatics and Bioengineering.* IEEE, 283–290.

Taslimitehrani, V., Dong, G., Pereira, N. L., Panahiazer, M., and Pathak, J. Developing EHR-driven Heart Failure Risk Prediction Models using CPXR(Log) with the Probabilistic Loss Function. In *Journal of Biomedical Informatics.* To appear.

Team, R. C. 2014. R: A language and environment for statistical computing. R Foundation for Statistical Computing.

Therneau, T. M., Atkinson, B., and Ripley, B. 2010. rpart: Recursive Partitioning. R package version 3.1-46. *Computer software program retrieved from http://CRAN. R-project. org/package= rpart.*

Thompson, D. 2015. High Cholesterol in Middle Age, Heart Risk Later? http://www.webmd.com/cholesterol–management/news/2.

Toms, J. D. and Lesperance, M. L. 2003. PIECEWISE REGRESSION: A TOOL FOR IDENTIFYING ECOLOGICAL THRESHOLDS. *Ecology 84,* 8 (aug), 2034–2041.

Trisha, G. 2014. *How to read a paper: The basics of evidence-based medicine.* John Wiley & Sons.

Tsanas, A. and Xifara, A. 2012. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings 49*, 560–567.

Urbanek, S. 2003. Rserve – A Fast Way to Provide R Functionality to Applications. *Proceedings of DSC 2*.

van Genuchten, M. T. 1980. A Closed-form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils1. *Soil Science Society of America Journal 44,* 5 (may), 892.

Vapnik, V. 2013. *The Nature of Statistical Learning Theory*. Springer Science & Business Media.

Wei, W., Li, J., Cao, L., Ou, Y., and Chen, J. 2013. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web 16,* 4, 449–475.

Yeh, I.-C. 1998. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research 28,* 12 (dec), 1797–1808.

Yin, X. and Han, J. 2003. CPAR: Classification based on Predictive Association Rules. In *SDM*. Vol. 3. SIAM, 369–376.

Yu, W., Liu, T., Valdez, R., Gwinn, M., and Khoury, M. J. 2010. Application of support vector machine modeling for prediction of common diseases: the case of diabetes and pre-diabetes. *BMC medical informatics and decision making 10,* 1 (jan), 16.

Zadrozny, B., Langford, J., and Abe, N. 2003. Cost-sensitive learning by cost-proportionate example weighting. *Third IEEE International Conference on Data Mining*, 435–442.

Zeileis, A., Hothorn, T., and Hornik, K. 2008. Model-based recursive partitioning. *Journal of Computational and Graphical Statistics 17,* 2, 492–514.

Zimmermann, A., Bringmann, B., and Rückert, U. 2010. Fast, effective molecular feature mining by local optimization. *Machine Learning and Knowledge Discovery in Databases*, 563–578.

Zimmermann, A. and Nijssen, S. 2014. Supervised pattern mining and applications to classification. *Frequent Pattern Mining*, 425–442.